IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Alexey D. Zinin

Title: PROTECTION OF NETWORK INFRASTRUCTURE AND SECURE COMMUNICATION OF CONTROL INFORMATION THERTO

App. No.: 10/782,390 Filed: 02-19-2004

Examiner: Chriss, Andrew W Group Art Unit: 2472

Atty. Docket No.: 1400.1376750

Mail Stop Appeal Brief - Patents
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

In response to the Notice of Appeal filed February 14, 2011, Appellant submits the following:

This appeal is taken under 35 U.S.C. § 134. A final Office action rejecting claims 1-38 has a mail date of September 8, 2010. A notice of appeal and a petition for extension of time were received at the United States Patent and Trademark Office on February 14, 2011. This appeal brief and a petition for extension of time were originally mailed June 14, 2011. A notification of non-compliant appeal brief was mailed July 11, 2011. This corrected appeal brief is being mailed July 20, 2011.

REAL PARTY IN INTEREST

As presently advised, Alcatel-Lucent Canada Inc.is the real party in interest in this appeal by virtue of an executed Assignment of the entire interest from the named Inventor(s), Alexey D. Zinin, to Alcatel-Lucent Canada Inc. recorded in the United States Patent and Trademark Office on June 10, 2004 at Reel 015457, Frame 0050, followed by a Certificate of Amalgamation from Alcatel Canada Inc. to Alcatel-Lucent Canada Inc. dated January 1, 2007. Appellant encloses copies of the above-referenced Assignment and Certificate of Amalgamation.

RELATED APPEALS AND INTERFERENCES

As presently advised, there are no other prior or pending appeals, interferences, or judicial proceedings known to Appellant, the Appellant's legal representative, or Assignee which may be related to, directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal.

STATUS OF CLAIMS

Claims 1-38 are pending in the present application. Claims 1-38 are finally rejected, the rejection of which is being appealed.

STATUS OF AMENDMENTS

Appellant has not amended the specification, drawings, or claims subsequent to final rejection. However, Appellant filed a response to the final rejection without amendment, for which an advisory action that did not change the status of the claims was received.

SUMMARY OF CLAIMED SUBJECT MATTER

1. (Previously Presented) A method for communicating Layer-3 control information in a communications network comprising the steps of:

marking packets carrying the Layer-3 control information {**as described, for example, on page 9, paragraph [0024], of the specification and illustrated, for example, by 215 of Fig. 2B**};

encapsulating the packets at Layer-2 to uniquely identify Layer-2 frames as carrying trusted control information {**as described, for example, on page 9, paragraph [0024], of the specification and illustrated, for example, by 218 of Fig. 2B**}.

2. (Original) The method of claim 1 wherein the step of marking further comprises:

marking the packets using a unique protocol identifier {**as described, for example, on page 9, paragraph [0024], of the specification and illustrated, for example, by 216 of Fig. 2B**}.

3. (Original) The method of claim 1 wherein the step of marking further comprises:

marking the packets using a link-local MPLS label {**as described, for example, on page 9, paragraph [0024], of the specification and illustrated, for example, by 217 of Fig. 2B**}.

4. (Original) The method of claim 1 further comprising the step of:

applying interface groups to determine when marking of control packets is to be done {**as described, for example, on page 8, paragraph [0021], of the specification and illustrated, for example, by 201 of Fig. 2A**}.

5. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:

applying interface groups to packet communications within a particular interface group {**as described, for example, on page 8, paragraph [0021], of the specification and illustrated, for example, by 203 of Fig. 2A**}.

6. (Original) The method of claim 5 wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of:

applying interface groups to packet communications within a backbone interface group {**as described, for example, on page 8, paragraph [0021], of the specification and illustrated, for example, by 204 of Fig. 2A**}.

7. (Original) The method of claim 5 wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of:

applying interface groups to packet communications within a customer-specific interface group {**as described, for example, on page 8, paragraph [0021], of the specification and illustrated, for example, by 205 of Fig. 2A**}.

8. (Original) The method of claim 5 wherein the step of applying interface groups

to packet communications within a particular interface group further comprises the step of:

applying interface groups to packet communications within a peer interface group {**as described, for example, on page 8, paragraph [0021], of the specification and illustrated, for example, by 206 of Fig. 2A**}.

9. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:

applying interface groups to packet communications between interface groups {**as described, for example, on page 8, paragraph [0022], of the specification and illustrated, for example, by 201 of Fig. 2A**}.

10. (Original) The method of claim 9 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between backbone and customer-specific interface groups {**as described, for example, on page 8, paragraph [0022], of the specification and illustrated, for example, by 208 of Fig. 2A**}.

11. (Original) The method of claim 9 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between customer-specific and peer interface groups {**as described, for example, on page 8, paragraph [0022], of the specification and illustrated, for example, by 209 of Fig. 2A**}.

12. (Original) The method of claim 9 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between backbone and peer interface groups {**as described, for example, on page 8, paragraph [0022], of the specification and illustrated, for example, by 210 of Fig. 2A**}.

13. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of ICMP packets {**as described, for example, on page 9, paragraph [0023], of the specification and illustrated, for example, by 211 of Fig. 2B**}.

14. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of ping packets {**as described, for example, on page 9, paragraph [0023], of the specification and illustrated, for example, by 212 of Fig. 2B**}.

15. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of traceroute packets {**as described, for example, on page 9, paragraph [0023], of the specification and illustrated, for example, by 213 of Fig. 2B**}.

16. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of packets from Network Operations Center (NOC) hosts {**as described, for example, on page 9, paragraph [0023], of the specification and illustrated, for example, by 214 of Fig. 2B**}.

17. (Original) The method of claim 1 wherein the step of encapsulating the packets further comprises:

encapsulating the packets according to control encapsulation {**as described, for example, on page 9, paragraph [0027], page 10, paragraph [0029], page 20, paragraph [0062], and/or page 21, paragraph [0063], of the specification and illustrated, for example, by control packet 123 of Fig. 1**}.

18. (Original) The method of claim 1 further comprising:

receiving unmarked control packets using rate-limited queues {**as described, for example, on page 9, paragraph [0027], of the specification**}.

19. (Original) The method of claim 1 further comprising:

receiving the packets as received packets; and

processing the received packets at a line rate {**as described, for example, on page 11, paragraph [0032], of the specification**}.

20. (Previously Presented) An apparatus comprising a network element {**as described, for example, on page 31, paragraph [0107] of the specification and illustrated, for example, by 106 of Fig. 1**} for communicating Layer-3 control information in a communications network {**as described, for example, on page 31, paragraphs [0016] and [0107] of the specification and illustrated, for example, by 104, 118, and 108 of Fig. 1**} adapted to perform the steps of:

marking packets carrying the Layer-3 control information {**as described, for example, on page 9, paragraph [0024], of the specification and illustrated, for example, by 215 of Fig. 2B**};

encapsulating the packets at Layer-2 to uniquely identify Layer-2 frames as carrying trusted control information {**as described, for example, on page 9, paragraph [0024], of the specification and illustrated, for example, by 218 of Fig. 2B**}.

21. (Original) The apparatus of claim 20 wherein the step of marking further comprises:

marking the packets using a unique protocol identifier {**as described, for example, on page 9, paragraph [0024], of the specification and illustrated, for example, by 216 of Fig. 2B**}.

22. (Original) The apparatus of claim 20 wherein the step of marking further comprises:

marking the packets using a link-local MPLS label {**as described, for example, on page 9, paragraph [0024], of the specification and illustrated, for example, by 217 of Fig. 2B**}.

23. (Original) The apparatus of claim 20 wherein the network element is further adapted to perform the step of:

applying interface groups to determine when marking of control packets is to be done {**as described, for example, on page 8, paragraph [0021], of the specification and illustrated, for example, by 201 of Fig. 2A**}.

24. (Original) The apparatus of claim 23 wherein the step of applying interface

groups further comprises the step of:

applying interface groups to packet communications within a particular interface group {**as described, for example, on page 8, paragraph [0021], of the specification and illustrated, for example, by 203 of Fig. 2A**}.

25. (Original) The apparatus of claim 24 wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of:

applying interface groups to packet communications within a backbone interface group {**as described, for example, on page 8, paragraph [0021], of the specification and illustrated, for example, by 204 of Fig. 2A**}.

26. (Original) The apparatus of claim 24 wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of:

applying interface groups to packet communications within a customer-specific interface group {**as described, for example, on page 8, paragraph [0021], of the specification and illustrated, for example, by 205 of Fig. 2A**}.

27. (Original) The apparatus of claim 24 wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of:

applying interface groups to packet communications within a peer interface group {**as described, for example, on page 8, paragraph [0021], of the specification and illustrated, for example, by 206 of Fig. 2A**}.

28. (Original) The apparatus of claim 23 wherein the step of applying interface groups further comprises the step of:

applying interface groups to packet communications between interface groups {**as described, for example, on page 8, paragraph [0022], of the specification and illustrated, for example, by 207 of Fig. 2A**}.

29. (Original) The apparatus of claim 28 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between backbone and customer-specific interface groups {**as described, for example, on page 8, paragraph [0022], of the specification and illustrated, for example, by 208 of Fig. 2A**}.

30. (Original) The apparatus of claim 28 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between customer-specific and peer interface groups {**as described, for example, on page 8, paragraph [0022], of the specification and illustrated, for example, by 209 of Fig. 2A**}.

31. (Original) The apparatus of claim 28 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between backbone and peer interface groups {**as described, for example, on page 8, paragraph [0022], of the specification and illustrated, for example, by 210 of Fig. 2A**}.

32. (Original) The apparatus of claim 23 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of ICMP packets {**as described, for example, on page 9, paragraph [0023], of the specification and illustrated, for example, by 211 of Fig. 2B**}.

33. (Original) The apparatus of claim 23 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of ping packets {**as described, for example, on page 9, paragraph [0023], of the specification and illustrated, for example, by 212 of Fig. 2B**}.

34. (Original) The apparatus of claim 23 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of traceroute packets {**as described, for example, on page 9, paragraph [0023], of the specification and illustrated, for example, by 213 of Fig. 2B**}.

35. (Original) The apparatus of claim 23 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of packets from Network Operations Center (NOC) hosts {**as described, for example, on page 9, paragraph [0023], of the specification and illustrated, for example, by 214 of Fig. 2B**}.

36. (Original) The apparatus of claim 20 wherein network element is further adapted to encapsulate the packets according to control encapsulation {**as described, for**

example, on page 9, paragraph [0027], page 10, paragraph [0029], page 20, paragraph [0062], and/or page 21, paragraph [0063], of the specification and illustrated, for example, by control packet 123 of Fig. 1}.

37. (Original) The method of claim 20 wherein the network element is further adapted to receive unmarked control packets using rate-limited queues {**as described, for example, on page 9, paragraph [0027], of the specification**}.

38. (Original) The apparatus of claim 20 wherein the network element is further adapted to receive the packets as received packets and to process the received packets at a line rate {**as described, for example, on page 11, paragraph [0032], of the specification**}.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to be reviewed on appeal are as follow:

The First Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 1, 2, 4, 17, 20, 21, 23 and 36 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1).

The Second Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 3 and 22 under 35 U.S.C. 103(a) as being unpatentable over McDysan (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 1 and 20 above, and further in view of Nakamichi et al (U.S. Patent Application Publication US 2002/0085498 A1).

The Third Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 5-12 and 24-31 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 4 and 23 above, and further in view of Yu et al. (United States Patent Application Publication US 2004/0010583 A1).

The Fourth Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 13 and 32 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 4 and 23 above, and further in view of Holden et al. (United States Patent No. 5,802,178).

The Fifth Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 14 and 33 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 4 and 23 above, and further in view of Pan et al. (United States Patent 7,336,615).

The Sixth Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 15 and 34 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent No. 7,046,680) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 4 and 23 above, and further in view of Fotedar et al. (United States Patent Application Publication 2004/0085965 A1).

The Seventh Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 16 and 35 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US

2002/0067725 A1) as applied to claims 4 and 23 above, and further in view of Tuomenoksa et al. (United States Patent Application Publication 2002/0023210 A1).

The Eighth Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 18 and 37 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 1 and 20 above, and further in view of Johansson (United States Patent 6,061,330).

The Ninth Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 19 and 38 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 1 and 20 above, and further in view of Hussey et al. (United States Patent Application Publication 2001/0049744 A1).

ARGUMENT

Claims 1-38 are pending in the application. The Examiner has rejected claims 1-38. Appellant appeals the rejection of claims 1-38.

Appellant submits there exist clear errors in the Examiner's rejections and/or the Examiner's omissions of one or more essential elements needed for a *prima facie* rejection. Appellant submits the Examiner's "Response to Arguments" provides evidence that the Examiner has failed to consider the pending claims as required by the Manual of Patent Examining Procedure (MPEP) and prevailing case law. MPEP § 2141 sets forth the Graham inquiries for a rejection under 35 U.S.C. § 103. *Graham v. John Deere*, 383 U.S. 1, 148 USPQ 459 (1966). MPEP § 2143 describes examples of basic requirements of a *prima facie* case of obviousness under 35 U.S.C. § 103. As Appellant describes in detail below, Appellant submits there exist clear errors in the Examiner's rejections and/or the Examiner's omissions of one or more essential elements needed for a *prima facie* rejection.

MPEP § 2141 states, in part, as follows:

> When applying 35 U.S.C. 103, the following tenets of patent law must be adhered to:
>
> (A) The claimed invention must be considered as a whole;
>
> (B) The references must be considered as a whole and must suggest the desirability and thus the obviousness of making the combination;
>
> (C) The references must be viewed without the benefit of impermissible hindsight vision afforded by the claimed invention; and
>
> (D) Reasonable expectation of success is the standard with which obviousness is determined.

Hodosh v. Block Drug Co., Inc., 786 F.2d 1136, 1143 n.5, 229 USPQ 182, 187 n.5 (Fed. Cir. 1986).

Appellant submits the Examiner has failed to consider the claimed subject matter as a whole but rather makes allegations with respect to one or more aspects of the claimed subject matter that contradict allegations the Examiner makes with respect to one or more other aspects of the claimed subject matter, as Appellant details below. Appellant submits the Examiner appears to fail to consider the references as a whole, ignoring, for example, portions of the reference that "teach away," and thereby teach against the desirability and thus the obviousness of making the combination, as Appellant details below. Appellant submits the Examiner appears to apply impermissible hindsight vision, attempting to contort the alleged teachings of the cited portions of the cited references to yield a modification that Appellant shows below, in many instances, to be inoperable. Appellant submits the Examiner does not appear to meet the reasonable expectation of success standard but rather fails to provide a plausible rationale as to how the cited references could allegedly work together, as Appellant details below. Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to the claimed subject matter.

For the Examiner to establish a *prima facie* case of obviousness under 35 U.S.C. § 103(a), Appellant submits the Examiner must, as a minimum, show that the claimed inventions as a whole (i.e. each and every element of the claimed invention in the claimed configuration) must have been obvious to one of skill in the art at the time the invention was made. MPEP § 2142. The Federal Circuit has stated that "rejections on obviousness cannot be sustained with mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness." *In re Kahn*, 441 F.3d 977, 988, 78 USPQ2d 1329, 1336 (Fed. Cir. 2006). As shown below, the Examiner's rejection of the claims fails to take into account each and every element of the claimed invention and fails to show that the

claimed invention as a whole would be obvious to one of ordinary skill in the art at the time the invention was made. Accordingly, the Examiner fails to establish a *prima facie* case of obviousness.

The First Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 1, 2, 4, 17, 20, 21, 23 and 36 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1). Appellant respectfully disagrees.

Claim 1:

Regarding claim 1, Appellant submits the cited portions of the cited references do not render obvious the subject matter of claim 1. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "marking packets carrying the Layer-3 control information." While the Examiner states, "McDysan discloses marking packets carrying Layer-3 control information (paragraphs 0037 and 0042, wherein packets are marked with a differentiated services code point (DSCP) value)," Appellant submits none of the cited portions of the cited references appear to teach or suggest, for example, "Layer-3 control information." Moreover, Appellant submits the Examiner's allegation of "which is known in the art as an implementation of 'Layer-3' in the OSI 7-layer Interconnect Model (i.e., the network layer)" does not appear to allege teaching as to, for example, "Layer-3 control information." Moreover, it isn't clear what noun or noun phrase the Examiner is using the word "which" to refer back to in alleging "which is known in the art as an implementation of 'Layer-3' in the OSI 7-layer Interconnect Model (i.e., the network layer)." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claim 1.

Also, Appellant notes paragraph [0042] of the McDysan (US 2003/0112755 A1) reference states, in part, "Marker M0 remarks all packets received at LP-2 110b with DSCP 0000, thus identifying the packets as best-effort traffic." Appellant submits "Marker M0 remarks all packets received..." fails to disclose or suggest, and teaches away from, "marking packets carrying the Layer-3 control information." Therefore, Appellant submits claim 1 is in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 1 and 20, the Examiner states as follows:

> Regarding Claims 1 and 20, Applicant states "none of the cited portions of the cited references appear to teach or suggest, for example, "Layer-3 control information." Examiner respectfully disagrees. Examiner notes that Applicant has not specifically pointed out how the language of the claims patentably distinguishes them from the references. Examiner turns to Applicant's specification at paragraph 0004, which defines the OSI 7-layer model. More specifically, the network layer is defined as Layer-3. Examiner further turns to The OSI Reference Model, which was provided to Applicant with the Office Action mailed June 5, 2009, wherein Layer-3 (i.e., the Network Layer) corresponds to the Internet Protocol (IP). Turning to the McDysan reference, paragraph 0009 recites: "Diffsery is an IP QoS architecture that achieves scalability by conveying an aggregate traffic classification within a DS field (e.g., the IPv4 Type of Service (TOS) byte or IPv6 traffic class byte) of each IP-layer packet header. The first six bits of the DS field encode a Diffsery Code Point (DSCP) that requests a specific class of service or Per Hop Behavior (PHB) for the packet at each node along its path within a Diffsery domain.." Examiner further notes that the claimed "control information" is not further defined in the claim language so as to require a structure or feature of said information other than being "Layer-3 control information." As the DSCP disclosed in McDysan controls the QoS applied to a packet (e.g., in paragraphs 0037 and 0042) and further is indicative of an IP QoS (i.e., Layer-3), Examiner submits that the claim limitation "Layer-3 control information" is met by the disclosure of McDysan. Applicant further states paragraph 0042 of McDysan "fails to disclose or suggest, and teaches away from "marking packets carrying the Layer-3 control information."
>
> Examiner respectfully disagrees. Examiner notes that while alleging that the disclosure of McDysan teaches away from marking packets, Applicant has not specifically pointed out how the disclosure teaches away or how the language of the claims patentably distinguishes them from the references. As described above, McDysan discloses marking packets via a DSCP code point in IP packets, and therefore meets the claim limitation "marking packets carrying the Layer-3 control information."

Appellant notes the Examiner alleges "...McDysan discloses marking packets via a DSCP code point in IP packets, and therefore meets the claim limitation 'marking packets carrying the Layer-3 control information.'" However, Appellant notes the Examiner also alleges, "As the DSCP disclosed in McDysan controls the QoS applied to a packet (e.g., in paragraphs 0037 and 0042) and further is indicative of an IP QoS (i.e., Layer-3), Examiner submits that the claims limitation "Layer-3 control information" is met by the disclosure of McDysan." However, if the Examiner is alleging that "the DSCP disclosed in McDysan" is the basis for the Examiner's contention "that the claims limitation 'Layer-3 control information' is met by the disclosure of McDysan," then Appellant submits the Examiner has not shown any portion of the cited references to disclose or suggest "...marking packets carrying the Layer-3 control information." However, if the Examiner is alleging that "...McDysan discloses marking packets via a DSCP code point in IP packets...," then Appellant submits the Examiner has not shown any portion of the cited references to disclose or suggest "...packets carrying the Layer-3 control information." Thus, Appellant submits the Examiner has not shown any portion of the cited references to disclose or suggest, for example, "...marking packets carrying the Layer-3 control information."

As another example, Appellant submits the Examiner has not alleged teaching as to "...encapsulating the packets at Layer-2 to uniquely identify Layer-2 frames as carrying trusted control information." While the Examiner cites "(paragraph 0215, Figure 25, wherein a packet containing L2TP is encapsulated with a PPP or Ethernet header)" and alleges teaching as to "...an implementation of 'Layer 2' of the OSI 7-layer Interconnect Model (i.e., the data link layer)," Appellant submits the Examiner does not allege teaching of "...to uniquely identify Layer-2 frames as carrying trusted control information." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to the subject matter of claims 1 or 20. Therefore, Appellant submits claims 1 and 20 are in condition for allowance.

Appellant submits the Court of Appeals for the Federal Circuit has also found "teaching away" to support a finding of non-obviousness even after *KSR*. For example, in *Crocs, Inc. v. ITC* (Fed. Cir. 2010), the Court found a footwear piece having "a strap section formed of a molded foam material" to be non-obvious where "the prior art showed that foam was an unsuitable material for shoe straps." In *Ricoh Co. v. Quanta Computer Inc.* (Fed. Cir. 2008), the Court stated, "A reference may be said to teach away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant."

Also, the Board of Patent Appeals and Interferences has, also in the post-*KSR* time frame, stated, "It is also a legal principle that when the prior art teaches away from combining certain known elements, discovery of a successful means of combining them is more likely to be nonobvious." *In re Ikeda*, Appeal 2008-0492 (March 26, 2008) at p. 5. The Board also stated, "...[W]hen the prior art teaches away from the claimed solution..., obviousness cannot be proven merely by showing that a known composition could have been modified by routine experimentation or solely on the expectation of success...." *Ex parte Whalen*, Appeal 2007-4423 (July 23, 2008) at p. 16. Thus, Appellant submits the Examiner's proposition as to allegedly establishing obviousness must be understood in the context of case law and Office practice that recognizes that evidence of "teaching away" undermines such an allegation of obviousness. Accordingly, Appellant submits the Examiner's rejection of claim 1 is improper.

Furthermore, while the Examiner alleges various teachings with respect to different references, even assuming arguendo the merit of the Examiner's allegations, Appellant submits "a patent composed of several elements is not proved obvious merely by demonstrating that each of its elements is, independently, known in the prior art." *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 405, 418 (2007). Appellant submits the

Examiner has not provided a plausible rationale as to why the prior art references would have worked together to render the claims obvious. *Power-One, Inc. v. Artesyn Techs., Inc.*, 599 F.3d 1343 (Fed. Cir. 2010). As an example, Appellant submits the Examiner's alleged a motivation as to "in order to find virtual routers on edge routers belonging to the same VPN so that the virtual routers belonging to the same VPN can be mutually connected with tunnels (such as the L2TP tunnel or the IPsec tunnel) in case routing information is exchanged between the virtual routers belonging to the same VPN (see paragraph 0085 of Oguchi)" fails to provide motivation as to the alleged use of Layer-3 marking, where the Examiner alleges teaching as to "packets are marked with differentiated services code point (DSCP) value." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claim 1.

Moreover, as Appellant argues above, Appellant submits the Examiner's alleged motivation is not looking only at the problem Appellant was trying to solve, which Appellant argues is inconsistent with *KSR Int'l Co. v. Teleflex, Inc.*, 550 U.S. 398 (2007). Appellant submits a person of ordinary skill in the art at the time of the invention would only be led to prior art elements attempting to solve the same problem, which Appellant submits is not the case with the cited references. Accordingly, Appellant submits there would have been no reasonable expectation of success for one of ordinary skill in the art at the time of the invention in view of the alleged teachings of the cited references. Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claim 1.

For the foregoing reasons, Appellant submits the Examiner erred in rejecting claim 1. Thus, Appellant submits claim 1 is in condition for allowance.

Claim 2:

Regarding claim 2, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claim 2. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "...wherein the step of marking further comprises: marking the packets using a unique protocol identifier." While the Examiner cites "(paragraph 0037 and 0042, wherein packets are marked with a three bit differentiated services code point (DSCP) value (e.g., 000, 010, and 101)," Appellant submits the Examiner does not explain how the Examiner considers "a three bit differentiated services code point (DSCP) value (e.g., 000, 010, and 101)" to be "a unique protocol identifier." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claim 2. Therefore, Appellant submits claim 2 is in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 2 and 21, the Examiner states as follows:

> Regarding Claims 2 and 21, Applicant states "Examiner does not explain how the Examiner considers "a three bit differentiated services code point value ... to be a "unique protocol identifier." Examiner respectfully disagrees. Examiner notes that the DSCP value disclosed in McDysan uniquely identifies how the packet is to be treated (e.g., binary value of 000 indicating the packet is to be treated as best-effort in paragraph 0042).

Appellant notes paragraph [0042] states, in relevant part, "...DSCP 000, thus identifying the packets as best-effort traffic." Thus, Appellant submits the "DSCP 000" does not disclose or suggest a "unique protocol identifier." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 2 and 21. Therefore, Appellant submits claims 2 and 21 are in condition for allowance.

Claim 4:

Regarding claim 4, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claim 4. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "applying interface groups to determine when marking of control packets is to be done." While the Examiner cites "(Figure 5 and paragraph 0036, wherein the classifier in the LAN port determines by reference to a classifier table indexed by multiple indices, such as source port and destination port, to determine an interface for communication and to send values to a packet marker)," Appellant submits the alleged teaching of "to send values to a packet marker" does not teach or suggest "...to determine when marking of control packets is to be done." Appellant submits the cited portions of the cited reference do not disclose or suggest "...to determine when marking of control packets is to be done." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claim 4. Therefore, Appellant submits claim 4 is in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 4 and 23, the Examiner states as follows:

> Regarding Claims 4 and 23, Applicant states "the alleged teaching of "to send values to a packet marker" does not teach or suggest "...to determine when marking of control packets is to be done." Examiner respectfully disagrees. Examiner notes that Applicant has not specifically pointed out how the language of the claims patentably distinguishes them from the references. McDysan, at Figure 5 and paragraph 0036, discloses a classifier in the LAN port determining, via by reference to a classifier table indexed by multiple indices (e.g., source port and destination port), to determine an interface for communication and to send values to a packet marker. Further, at paragraphs 0037 and 0042, a determination is made with regard to marking of a packet (e.g., marking a packet when received from an access network).

Appellant submits none of the portions of the cited references appear to teach or suggest "to determine when marking of control packets is to be done." Rather,

Appellant submits "...classifier 124 of LP-2 110b directs all packets to marker M0 in accordance with classifier table 126" and "Marker M0 remarks all packets received at LP-2 110b with DSCP 000, thus identifying the packets as best-effort traffic" of paragraph [0042] teach away from "determine when marking of control packets is to be done," as they teach remarking of all received packets indiscriminately. Also, Appellant notes the Examiner alleges teaching only as to "...marking of a packet...," not "...marking of control packets...." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 4 and 23. Therefore, Appellant submits claims 4 and 23 are in condition for allowance.

Claim 17:

Regarding claim 17, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claim 17. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of encapsulating the packets further comprises: encapsulating the packets according to control encapsulation." While the Examiner cites "(paragraph 0215, Figure 25, wherein a packet containing an IP header)" in the Oguchi reference, Appellant submits the cited portions of the cited reference do not appear to disclose, as an example, "...according to control encapsulation." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claim 17. Therefore, Appellant submits claim 17 is in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 17 and 36, the Examiner states as follows:

> Regarding Claims 17 and 36, Applicant states "the cited portions of the cited reference do not appear to disclose, as an example, according to control encapsulation." Examiner respectfully disagrees. Examiner notes that Applicant has not specifically pointed out how the language of the claims patentably distinguishes them from the references.

> Further, Examiner notes that the claimed "control encapsulation" is not further defined in the claim language so as to require a certain format for the encapsulation. As such, Examiner gives the claim language its broadest reasonable interpretation without unnecessarily importing limitations from the specification. Oguchi discloses encapsulating an L2TP VPN packet (i.e., performing control encapsulation) comprising Layer 3 encapsulation (paragraph 0215, Figure 25, wherein a packet containing an IP header).

While the Examiner states "'control encapsulation' is not further defined in the claim language so as to require a certain format for the encapsulation," Appellant submits the Examiner has not offered any explanation as to how "encapsulating an L2TP VPN packet" discloses or suggests "control encapsulation." Rather, the Examiner parenthetically states, "(i.e., performing control encapsulation)" without any explanation or justification. Moreover, Appellant submits paragraph [0215], as cited by the Examiner, merely states "It is to be noted that the present embodiment has dealt with the case where the L2TP tunneling is used as a tunneling technique" and "The format of the encapsulated packet transmitted through the L2TP tunnel in such case is the same as that shown in FIG. 25." Appellant submits such teaching fails to disclose or suggest "control encapsulation." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 17 and 36. Therefore, Appellant submits claims 17 and 36 are in condition for allowance.

Claim 20:

Regarding claim 20, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claim 20. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "marking packets carrying the Layer-3 control information." While the Examiner states, "McDysan discloses apparatus comprising a network element (Figure 5, CPE edge router 34 comprising LAN physical ports (60a-60n) and WAN physical ports 64a-64n that further comprise packet classifiers 80 (LAN) and 100 (WAN)) that marks packets carrying

Layer-3 control information (paragraphs 0037 and 0042, wherein packets are marked with a differentiated services code point (DSCP) value)," Appellant submits none of the cited portions of the cited references appear to teach or suggest, for example, "Layer-3 control information." Moreover, Appellant submits the Examiner's allegation of "which is known in the art as an implementation of 'Layer-3' in the OSI 7-layer Interconnect Model (i.e., the network layer)" does not appear to allege teaching as to, for example, "Layer-3 control information." Moreover, it isn't clear what noun or noun phrase the Examiner is using the word "which" to refer back to in alleging "which is known in the art as an implementation of 'Layer-3' in the OSI 7-layer Interconnect Model (i.e., the network layer)." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claim 20.

Also, Appellant notes paragraph [0042] of the McDysan (US 2003/0112755 A1) reference states, in part, "Marker M0 remarks all packets received at LP-2 110b with DSCP 0000, thus identifying the packets as best-effort traffic." Appellant submits "Marker M0 remarks all packets received..." fails to disclose or suggest, and teaches away from, "marking packets carrying the Layer-3 control information." Therefore, Appellant submits claim 20 is in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 1 and 20, the Examiner states as follows:

> Regarding Claims 1 and 20, Applicant states "none of the cited portions of the cited references appear to teach or suggest, for example, "Layer-3 control information." Examiner respectfully disagrees. Examiner notes that Applicant has not specifically pointed out how the language of the claims patentably distinguishes them from the references. Examiner turns to Applicant's specification at paragraph 0004, which defines the OSI 7-layer model. More specifically, the network layer is defined as Layer-3. Examiner further turns to The OSI Reference Model, which was provided to Applicant with the Office Action mailed June 5, 2009, wherein Layer-3 (i.e., the Network Layer) corresponds to the Internet Protocol (IP). Turning to the McDysan reference, paragraph 0009 recites: "Diffsery is an IP QoS architecture that achieves scalability by conveying an aggregate traffic classification within a DS field (e.g., the IPv4 Type of Service (TOS) byte or IPv6 traffic class byte) of each IP-layer packet

> header. The first six bits of the DS field encode a Diffsery Code Point (DSCP) that requests a specific class of service or Per Hop Behavior (PHB) for the packet at each node along its path within a Diffsery domain.." Examiner further notes that the claimed "control information" is not further defined in the claim language so as to require a structure or feature of said information other than being "Layer-3 control information." As the DSCP disclosed in McDysan controls the QoS applied to a packet (e.g., in paragraphs 0037 and 0042) and further is indicative of an IP QoS (i.e., Layer-3), Examiner submits that the claim limitation "Layer-3 control information" is met by the disclosure of McDysan. Applicant further states paragraph 0042 of McDysan "fails to disclose or suggest, and teaches away from "marking packets carrying the Layer-3 control information."
>
> Examiner respectfully disagrees. Examiner notes that while alleging that the disclosure of McDysan teaches away from marking packets, Applicant has not specifically pointed out how the disclosure teaches away or how the language of the claims patentably distinguishes them from the references. As described above, McDysan discloses marking packets via a DSCP code point in IP packets, and therefore meets the claim limitation "marking packets carrying the Layer-3 control information."

Appellant notes the Examiner alleges "...McDysan discloses marking packets via a DSCP code point in IP packets, and therefore meets the claim limitation 'marking packets carrying the Layer-3 control information.'" However, Appellant notes the Examiner also alleges, "As the DSCP disclosed in McDysan controls the QoS applied to a packet (e.g., in paragraphs 0037 and 0042) and further is indicative of an IP QoS (i.e., Layer-3), Examiner submits that the claims limitation "Layer-3 control information" is met by the disclosure of McDysan." However, if the Examiner is alleging that "the DSCP disclosed in McDysan" is the basis for the Examiner's contention "that the claims limitation 'Layer-3 control information' is met by the disclosure of McDysan," then Appellant submits the Examiner has not shown any portion of the cited references to disclose or suggest "...marking packets carrying the Layer-3 control information." However, if the Examiner is alleging that "...McDysan discloses marking packets via a DSCP code point in IP packets...," then Appellant submits the Examiner has not shown any portion of the cited references to disclose or suggest "...packets carrying the Layer-3 control information." Thus, Appellant submits the Examiner has not shown any portion of the cited references to disclose or suggest, for example, "...marking packets carrying the Layer-3 control information."

As another example, Appellant submits the Examiner has not alleged teaching as to "...encapsulating the packets at Layer-2 to uniquely identify Layer-2 frames as carrying trusted control information." While the Examiner cites "(paragraph 0215, Figure 25, wherein a packet containing L2TP is encapsulated with a PPP or Ethernet header)" and alleges teaching as to "...an implementation of 'Layer 2' of the OSI 7-layer Interconnect Model (i.e., the data link layer)," Appellant submits the Examiner does not allege teaching of "...to uniquely identify Layer-2 frames as carrying trusted control information." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to the subject matter of claims 1 or 20. Therefore, Appellant submits claims 1 and 20 are in condition for allowance.

Claim 21:

Regarding claim 21, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claim 21. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of marking further comprises: marking the packets using a unique protocol identifier." While the Examiner cites "(paragraph 0037 and 0042, wherein packets are marked with a three bit differentiated services code point (DSCP) value (e.g., 000, 010, and 101)," Appellant submits the Examiner does not explain how the Examiner considers "a three bit differentiated services code point (DSCP) value (e.g., 000, 010, and 101)" to be "a unique protocol identifier." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claim 21. Therefore, Appellant submits claim 21 is in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 2 and 21, the Examiner states as follows:

> Regarding Claims 2 and 21, Applicant states "Examiner does not explain how the Examiner considers "a three bit differentiated services code point value ... to be a "unique protocol identifier." Examiner respectfully disagrees. Examiner notes that the DSCP value disclosed in McDysan uniquely identifies how the packet is to be treated (e.g., binary value of 000 indicating the packet is to be treated as best-effort in paragraph 0042).

Appellant notes paragraph [0042] states, in relevant part, "...DSCP 000, thus identifying the packets as best-effort traffic." Thus, Appellant submits the "DSCP 000" does not disclose or suggest a "unique protocol identifier." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 2 and 21. Therefore, Appellant submits claims 2 and 21 are in condition for allowance.

Claim 23:

Regarding claim 23, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claim 23. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "...wherein the network element is further adapted to perform the step of: applying interface groups to determine when marking of control packets is to be done." While the Examiner cites "(Figure 5 and paragraph 0036, wherein the classifier in the LAN port determines by reference to a classifier table indexed by multiple indices, such as source port and destination port, to determine an interface for communication and to send values to a packet marker)," Appellant submits the alleged teaching of "to send values to a packet marker" does not teach or suggest "...to determine when marking of control packets is to be done." Appellant submits the cited portions of the cited reference do not disclose or suggest "...to determine when marking of control packets is to be done." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claim 23. Therefore, Appellant submits claim 23 is in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 4 and 23, the Examiner states as follows:

> Regarding Claims 4 and 23, Applicant states "the alleged teaching of "to send values to a packet marker" does not teach or suggest "...to determine when marking of control packets is to be done." Examiner respectfully disagrees. Examiner notes that Applicant has not specifically pointed out how the language of the claims patentably distinguishes them from the references. McDysan, at Figure 5 and paragraph 0036, discloses a classifier in the LAN port determining, via by reference to a classifier table indexed by multiple indices (e.g., source port and destination port), to determine an interface for communication and to send values to a packet marker. Further, at paragraphs 0037 and 0042, a determination is made with regard to marking of a packet (e.g., marking a packet when received from an access network).

Appellant submits none of the portions of the cited references appear to teach or suggest "to determine when marking of control packets is to be done." Rather, Appellant submits "...classifier 124 of LP-2 110b directs all packets to marker M0 in accordance with classifier table 126" and "Marker M0 remarks all packets received at LP-2 110b with DSCP 000, thus identifying the packets as best-effort traffic" of paragraph [0042] teach away from "determine when marking of control packets is to be done," as they teach remarking of all received packets indiscriminately. Also, Appellant notes the Examiner alleges teaching only as to "...marking of a packet...," not "...marking of control packets...." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 4 and 23. Therefore, Appellant submits claims 4 and 23 are in condition for allowance.

Claim 36:

Regarding claim 36, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claim 36. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein network element is further adapted to encapsulate the packets according to control encapsulation." While the Examiner cites "(paragraph 0215, Figure 25, wherein a

packet containing an IP header)" in the Oguchi reference, Appellant submits the cited portions of the cited reference do not appear to disclose, as an example, "...according to control encapsulation." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claim 36. Therefore, Appellant submits claim 36 is in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 17 and 36, the Examiner states as follows:

> Regarding Claims 17 and 36, Applicant states "the cited portions of the cited reference do not appear to disclose, as an example, according to control encapsulation." Examiner respectfully disagrees. Examiner notes that Applicant has not specifically pointed out how the language of the claims patentably distinguishes them from the references. Further, Examiner notes that the claimed "control encapsulation" is not further defined in the claim language so as to require a certain format for the encapsulation. As such, Examiner gives the claim language its broadest reasonable interpretation without unnecessarily importing limitations from the specification. Oguchi discloses encapsulating an L2TP VPN packet (i.e., performing control encapsulation) comprising Layer 3 encapsulation (paragraph 0215, Figure 25, wherein a packet containing an IP header).

While the Examiner states "'control encapsulation' is not further defined in the claim language so as to require a certain format for the encapsulation," Appellant submits the Examiner has not offered any explanation as to how "encapsulating an L2TP VPN packet" discloses or suggests "control encapsulation." Rather, the Examiner parenthetically states, "(i.e., performing control encapsulation)" without any explanation or justification. Moreover, Appellant submits paragraph [0215], as cited by the Examiner, merely states "It is to be noted that the present embodiment has dealt with the case where the L2TP tunneling is used as a tunneling technique" and "The format of the encapsulated packet transmitted through the L2TP tunnel in such case is the same as that shown in FIG. 25." Appellant submits such teaching fails to disclose or suggest "control encapsulation." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 17 and 36. Therefore, Appellant submits claims 17 and 36 are in condition for allowance.

The Second Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 3 and 22 under 35 U.S.C. 103(a) as being unpatentable over McDysan (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 1 and 20 above, and further in view of Nakamichi et al (U.S. Patent Application Publication US 2002/0085498 A1). Appellant respectfully disagrees.

Claims 3 and 22:

Regarding claims 3 and 22, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claims 3 and 22. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "...wherein the step of marking further comprises: marking the packets using a link-local MPLS label." While the Examiner states, "It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the link state advertisement disclosed in Nakamichi with the marker/policer disclosed in McDysan, as modified above, in order to allow a node in a communications network to collect traffic information and perform load sharing depending on traffic conditions," Appellant submits "to allow a node in a communications network to collect traffic information and perform load sharing depending on traffic conditions," Appellant submits such rationale would not have motivated one of ordinary skill in the art to combine the alleged teachings of the cited portions of the cited references so as to purportedly yield the subject matter of claims 3 and 22. Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 3 and 22. Therefore, Appellant submits claims 3 and 22 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 3 and 22, the Examiner states as follows:

> Regarding Claims 3 and 22, Applicant states ""to allow a node in a communications network to collect traffic information and perform load sharing depending on traffic conditions"...would not have motivated one of ordinary skill in the art to combine the alleged teachings of the cited portions of the cited references." Examiner respectfully disagrees. Examiner submits recognizes that obviousness may be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See In re Fine, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988), In re Jones, 958 F.2d 347,21 USPQ2d 1941 (Fed. Cir. 1992), and KSR International Co. v. Teleflex, Inc., 550 U.S. 398, 82 USPQ2d 1385 (2007). In this case, Nakamichi discloses, at paragraph 0012, a need to allow a node in a communication network to collect traffic information to thereby achieve load sharing depending on the conditions of the traffic. Therefore, Examiner notes that the references themselves provide a motivation to combine the disclosed teachings.

Appellant submits the Examiner has not shown how an alleged motivation of "a need to allow a node in a communication network to collect traffic information to thereby achieve load sharing depending on the conditions of the traffic" would have motivated one of ordinary skill in the art to combine the teachings of Nakamichi, directed to a device and method for collecting traffic information, with the teachings of McDysan, directed to a VPN-aware CPE edge router, and the teachings of Oguchi, directed to the establishment of virtual links between all of the relaying apparatuses belonging to a multicast address group, to allegedly yield marking packets carrying Layer-3 control information using a link-local MPLS label and encapsulating the packets at Layer-2 to uniquely identify Layer-2 frames as carrying trusted control information. Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 3 and 22. Therefore, Appellant submits claims 3 and 22 are in condition for allowance.

The Third Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 5-12 and 24-31 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US

2002/0067725 A1) as applied to claims 4 and 23 above, and further in view of Yu et al. (United States Patent Application Publication US 2004/0010583 A1). Appellant respectfully disagrees.

Claims 5 and 24:

Regarding claims 5 and 24, Appellant submits the cited portions of the cited reference fail to render obvious the subject matter of claims 5 and 24. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups further comprises the step of: applying interface groups to packet communications within a particular interface group." While the Examiner cites "(Figure 1, interface group defined between interfaces 'a' and 'd' within network device A)," Appellant submits the block diagram of Figure 1 of the Yu et al. reference does not disclose, as an example, "...the step of: applying interface groups...." As another example, Appellant submits the block diagram of Figure 1 of the Yu et al. reference does not disclose "applying interface groups to packet communications within a particular interface group." Contrary to the Examiner's assertion, Appellant submits "Figure 1" does not appear to disclose "interface group defined between interfaces 'a' and 'd' within network device A." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 5 and 24. Therefore, Appellant submits claims 5 and 24 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 5 and 24, the Examiner states as follows:

> Regarding Claims 5 and 24, Applicant states that "Figure 1" does not appear to disclose "interface group defined between interfaces 'a' and 'd' within network device A." Examiner respectfully disagrees. Claims 5 and 24 require "applying interface groups to packet communications within a particular interface group." However, Examiner notes that the claim language is not further defined so as to further limit the step of applying

> interface groups or the features of a particular interface group. Per MPEP 2106: "USPTO personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure. In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023, 1027-28 (Fed. Cir. 1997). Limitations appearing in the specification but not recited in the claim should not be read into the claim. E-Pass Techs., Inc. v. 3Com Corp., 343 F.3d 1364, 1369, 67 USPQ2d 1947, 1950 (Fed. Cir. 2003) (claims must be interpreted "in view of the specification" without importing limitations from the specification into the claims unnecessarily). In re Prater, 415 F.2d 1393, 1404-05, 162 USPQ 541, 550-551 (CCPA 1969). See also In re Zletz, 893 F.2d 319, 321-22, 13 USPQ2d 1320,1322 (Fed. Cir. 1989)." Examiner has given said claim language its broadest reasonable interpretation in view of the specification to comprise determination of an interface for communications. Accordingly, Yu discloses assigning interfaces to communicate within and between various types of networks (see Figures 1 and 4 and paragraphs 0022 and 0025). Further regarding Claims 5 and 24, Yu discloses packet communications between interfaces 'a' and `d' of Network Device A in Figure 1 (at paragraph 0034, wherein tunnel interface 'd' is assigned to physical interface 'a' within an interface group).

Appellant notes the Examiner does not appear to allege "applying interface groups..." Rather, Appellant notes the Examiner alleges, "Yu discloses assigning interfaces to communicate within and between various types of networks." Moreover, while paragraph [0034] of Yu states, "...define an interface group...," Appellant submits paragraph [0034] teaches doing so to "cause network device A to instruct network device B to assume mastership of the virtual IP address associated with the private physical interface 'a' of network device A, which will, in turn, cause network device B to assume mastership for that interface." Appellant submits the cited portion of the Yu reference does not teach or suggest "applying interface groups to determine when marking of control packets is to be done," comprising "applying interface groups to packet communications within a particular interface group," as Appellant submits instructing devices to assume mastership of a virtual IP address teaches away from "applying interface groups to determine when marking of control packets is to be done." Moreover, Appellant submits the Examiner has not shown how an alleged motivation of "to withstand failures of network device components, without triggering unnecessary failover in a network device" would have motivated one of ordinary skill in the art to combine the teachings of Yu, directed to a method and apparatus for defining failover

events in a network device, with the teachings of McDysan, directed to a VPN-aware CPE edge router, and the teachings of Oguchi, directed to the establishment of virtual links between all of the relaying apparatuses belonging to a multicast address group, to allegedly yield applying interface groups to packet communications within a particular interface group to determine when marking of control packets is to be done, marking packets carrying Layer-3 control information, and encapsulating the packets at Layer-2 to uniquely identify Layer-2 frames as carrying trusted control information. Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 5 and 24. Therefore, Appellant submits claims 5 and 24 are in condition for allowance.

Claims 6 and 25:

Regarding claims 6 and 25, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claims 6 and 25. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of: applying interface groups to packet communications within a backbone interface group." While the Examiner cites "(Figure 4, static tunnel through Internet between network device A and network device B)," Appellant submits the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest, as an example, "...the step of: applying interface groups to packet communications within a backbone interface group." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 6 and 25. Therefore, Appellant submits claims 6 and 25 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 6 and 25, the Examiner states as follows:

> Regarding Claims 6 and 25, Applicant states that "the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest, as an example, "...the step of. applying interface groups to packet communications within a backbone interface group." Examiner respectfully disagrees. Figure 4 of Yu discloses setting up a static tunnel (i.e., "Static Tunnel A") across the Internet (i.e., backbone) between two network devices. Given its broadest reasonable interpretation, the claimed "backbone interface group" limitation is met by interface 'd', which connects Network Device A. to the tunnel over the Internet.

Appellant notes the Examiner parenthetically characterizes "the Internet" as teaching "(i.e., backbone)," without citing any reference or providing any justification or explanation as to such characterization. Moreover, Appellant submits Figure 4 of Yu, as cited by the Examiner, does not appear to disclose "applying interface groups...." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 6 and 25. Therefore, Appellant submits claims 6 and 25 are in condition for allowance.

Claims 7 and 26:

Regarding claims 7 and 26, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claims 7 and 26. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of: applying interface groups to packet communications within a customer-specific interface group." While the Examiner cites "(Figure 4, interface 'a' between network device A and Host PC)," Appellant submits the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest "...the step of: applying interface groups to packet communications within a customer-specific interface group." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 7 and 26. Therefore, Appellant submits claims 7 and 26 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 7 and 26, the Examiner states as follows:

> Regarding Claims 7 and 26, Applicant states that "the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest, as an example, "...the step of: applying interface groups to packet communications within a customer-specific interface group." Examiner respectfully disagrees. Yu discloses communications with assigning interface 'a' to interconnect with a Host PC (i.e., customer-specific interface group given its broadest reasonable interpretation) in Figure 4.

Appellant submits Figure 4 of Yu, as cited by the Examiner, does not appear to disclose "applying interface groups...." Moreover, Appellant submits the Examiner does not appear to allege teaching as to "applying interface groups to packet communications within a customer-specific interface group." Rather, the Examiner alleges, "Yu discloses...assigning interface 'a' to interconnect with a Host PC." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 7 and 26. Therefore, Appellant submits claims 7 and 26 are in condition for allowance.

Claim 8 and 27:

Regarding claims 8 and 27, Appellant submits the cited portions of the cited references do not render obvious the subject matter of claims 8 and 27. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of: applying interface groups to packet communications within a peer interface group." While the Examiner cites "(Figure 4, static tunnel between network device A and network device D)," Appellant submits the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest "...the step of: applying interface groups to packet communications within a peer interface group." Thus, Appellant submits the Examiner has not made a *prima*

facie showing of obviousness with respect to claims 8 and 27. Therefore, Appellant submits claims 8 and 27 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 8 and 27, the Examiner states as follows:

> Regarding Claims 8 and 27, Applicant states that "the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest, as an example, "...the step of: applying interface groups to packet communications within a peer interface group." Yu discloses communications via a static tunnel between Network Device A and Network Device D (i.e., peer devices given its broadest reasonable interpretation) via interface 'a' on Network Device A (see Figure 4). Given its broadest reasonable interpretation, the claimed "peer interface group" limitation is met by the disclosed interface assignment (i.e., interface 'd') used in order to communicate between like devices (i.e., Network Device A and Network Device D).

Appellant submits Figure 4 of the Yu reference does not disclose or suggest "communications via a static tunnel between Network Device A and Network Device D (i.e., peer devices given its broadest reasonable interpretation) via interface 'a' on Network Device A (see Figure 4)," as alleged by the Examiner. Moreover, Appellant submits Figure 4 of the Yu reference does not disclose or suggest "(i.e., interface 'd') used in order to communicate between like devices (i.e., Network Device A and Network Device D)." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 8 and 27. Therefore, Appellant submits claims 8 and 27 are in condition for allowance.

Claim 9 and 28:

Regarding claims 9 and 28, Appellant submits the cited portions of the cited references do not render obvious the subject matter of claims 9 and 28. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups further comprises the step of: applying interface groups to packet communications between interface groups." While the

Examiner cites "(Figure 4, connections between peer, backbone, and customer networks at network device A)," Appellant submits the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest "...the step of: applying interface groups to packet communications between interface groups." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 9 and 28. Therefore, Appellant submits claims 9 and 28 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 9 and 28, the Examiner states as follows:

> Regarding Claims 9 and 28, Applicant states that "the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest, as an example, "...the step of: applying interface groups to packet communications between interface groups." Examiner respectfully disagrees.
>
> As stated above, Examiner has given the claim language "applying interface groups" its broadest reasonable interpretation in view of the specification to comprise determination of an interface for communications. Yu discloses applying interface groups to packet communications between interface groups (Figure 4, connections between peer (e.g., between Network Device A and Network Device D), backbone (e.g., in Network Device A between interfaces 'a' and `d', and customer networks (e.g., between Network Device A at interface 'a' and Host PC 12).

Appellant submits paragraph [0034] of the Yu reference appears to teach away from the Examiner's assertion that "Examiner has given the claim language 'applying interface groups' its broadest reasonable interpretation in view of the specification to comprise determination of an interface for communications." Appellant submits Yu's usage of "...define an interface group..." in paragraph [0034] is narrower than the Examiner's alleged "broadest reasonable interpretation." Thus, Appellant submits one of ordinary skill in the art at the time the invention was made, in view of the Yu reference, would not have understood "...define an interface group..." to merely mean "determination of an interface for communications," as alleged by the Examiner. Moreover, Appellant submits Figure 4 of the Yu reference, as cited by the Examiner, does not disclose or suggest "applying interface groups to packet communications

between interface groups." Furthermore, Appellant submits paragraph [0034] of the Yu reference does not disclose or suggest "applying interface groups to packet communications between interface groups (Figure 4, connections between peer (e.g., between Network Device A and Network Device D), backbone (e.g., in Network Device A between interfaces 'a' and `d', and customer networks (e.g., between Network Device A at interface 'a' and Host PC 12)." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 9 and 28. Therefore, Appellant submits claims 9 and 28 are in condition for allowance.

Claim 10 and 29:

Regarding claims 10 and 29, Appellant submits the cited portions of the cited references do not render obvious the subject matter of claims 10 and 29. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups to packet communications between interface groups further comprises the step of: applying interface groups to packet communications between backbone and customer-specific interface groups." While the Examiner cites "(Figure 4, connections between peer, backbone, and customer networks at network device A)," Appellant submits the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest "...the step of: applying interface groups to packet communications between backbone and customer-specific interface groups." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 10 and 29. Therefore, Appellant submits claims 10 and 29 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 10 and 29, the Examiner states as follows:

> Regarding Claims 10 and 29, Applicant states that "the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest, as an example, "...the step of: applying interface groups to packet communications between backbone and customer-specific interface groups." Examiner respectfully disagrees. Yu further discloses applying interface groups to packet communications between backbone and customer-specific groups (Figure 4, connections between backbone (e.g., in Network Device A between interfaces 'a' and `d' and customer networks (e.g., between Network Device A at interface 'a' and Host PC 12).

Appellant submits Figure 4 of the Yu reference does not disclose or suggest "applying interface groups to packet communications between backbone and customer-specific groups (Figure 4, connections between backbone (e.g., in Network Device A between interfaces 'a' and `d' and customer networks (e.g., between Network Device A at interface 'a' and Host PC 12)." Appellant notes paragraph [0034] of the Yu reference recites "...define an interface group containing the tunnel interface 'd.'" However, Appellant submits Yu teaches away from "applying interface groups" to "connections between backbone (e.g., in Network Device A between interfaces 'a' and `d' and customer networks (e.g., between Network Device A at interface 'a' and Host PC 12)," as Appellant submits such an alleged "applying interface groups" would appear to render inoperable the "tunnel fail over...without running a dynamic routing protocol" described in paragraph [0034] of the Yu reference. Accordingly, Appellant submits the Examiner fails to present a plausible rationale presented as to how the cited references could allegedly work together.

Appellant notes, even after *KSR International Co. v. Teleflex, Inc.*, 550 U S 398, 82 USPQ2d 1385 (2007), the Court of Appeals for the Federal Circuit has found inoperability to support a finding of non-obviousness. For example, in *DePuy Spine, Inc. v. Medtronic Sofamor Danek*, (Fed. Cir. June 1, 2009), the Court stated, "The 'predictable result' discussed in *KSR* refers not only to the expectation that prior art elements are capable of being physically combined, but also that the combination would have worked for its intended purpose," (citing *KSR*, at 1739-40) and "A reference teaches away from a combination when using it in that combination would produce an

inoperative result," (citing *In re ICON Health Fitness* (Fed. Cir. 2007). Furthermore, in *Power-One v. Artesyn* (Fed. Cir. March 30, 2010), the Court stated, "[A] patent composed of several elements is not proved obvious merely by demonstrating that each of its elements is, independently, known in the prior art." The Court could find no plausible rationale presented as to how the references could allegedly work together. Appellant submits the Board of Patent Appeals and Interferences continues to find inoperability to be persuasive of non-obviousness in a post-*KSR* context, for example, where the Board stated, "We are persuaded by the above arguments and agree with the Appellants that one of ordinary skill would not combine Forni and Heinz in the manner suggested by the Examiner because such a combination will result in an inoperable spring pad." *In re Thomas*, Appeal 2008-0173 (April 17, 2008). Thus, Appellant submits the apparent inoperability of the purported modification of the alleged teachings of the Mann reference must be understood in the context of case law and Office practice that recognizes that evidence of "inoperability" of the purported combination of the cited prior art references undermines such an allegation of obviousness. As Appellant alleges such "inoperability" as to the Examiner's purported combination, Appellant submits the Examiner's rejection of claims 10 and 29 is improper.

Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 10 and 29. Therefore, Appellant submits claims 10 and 29 are in condition for allowance.

Claim 11 and 30:

Regarding claims 11 and 30, Appellant submits the cited portions of the cited references do not render obvious the subject matter of claims 11 and 30. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups to packet communications

between interface groups further comprises the step of: applying interface groups to packet communications between customer-specific and peer interface groups." While the Examiner cites "(Figure 4, connections between peer, backbone, and customer networks at network device A)," Appellant submits the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest "...the step of: applying interface groups to packet communications between customer-specific and peer interface groups." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 11 and 30. Therefore, Appellant submits claims 11 and 30 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 11 and 30, the Examiner states as follows:

> Regarding Claims 11 and 30, Applicant states that "the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest, as an example, "...the step of. applying interface groups to packet communications between customer-specific and peer interface groups." Examiner respectfully disagrees. Yu discloses applying interface groups to packet communications between customer-specific and peer interface groups (Figure 4, connections between peer (e.g., between Network Device A and Network Device D) and customer networks (e.g., between Network Device A at interface 'a' and Host PC 12).

Appellant submits Figure 4 of the Yu reference does not disclose or suggest "applying interface groups to packet communications between customer-specific and peer interface groups (Figure 4, connections between peer (e.g., between Network Device A and Network Device D) and customer networks (e.g., between Network Device A at interface 'a' and Host PC 12)." Appellant notes paragraph [0034] of the Yu reference recites "...define an interface group containing the tunnel interface 'd.'" However, Appellant submits Yu teaches away from "applying interface groups" to "connections between peer (e.g., between Network Device A and Network Device D) and customer networks (e.g., between Network Device A at interface 'a' and Host PC 12)," as Appellant submits such an alleged "applying interface groups" would appear to render inoperable the "tunnel fail over...without running a dynamic routing protocol"

described in paragraph [0034] of the Yu reference. Accordingly, Appellant submits the Examiner fails to present a plausible rationale presented as to how the cited references could allegedly work together.

Appellant notes, even after *KSR International Co. v. Teleflex, Inc.*, 550 U S 398, 82 USPQ2d 1385 (2007), the Court of Appeals for the Federal Circuit has found inoperability to support a finding of non-obviousness. For example, in *DePuy Spine, Inc. v. Medtronic Sofamor Danek*, (Fed. Cir. June 1, 2009), the Court stated, "The 'predictable result' discussed in *KSR* refers not only to the expectation that prior art elements are capable of being physically combined, but also that the combination would have worked for its intended purpose," (citing *KSR*, at 1739-40) and "A reference teaches away from a combination when using it in that combination would produce an inoperative result," (citing *In re ICON Health Fitness* (Fed. Cir. 2007). Furthermore, in *Power-One v. Artesyn* (Fed. Cir. March 30, 2010), the Court stated, "[A] patent composed of several elements is not proved obvious merely by demonstrating that each of its elements is, independently, known in the prior art." The Court could find no plausible rationale presented as to how the references could allegedly work together. Appellant submits the Board of Patent Appeals and Interferences continues to find inoperability to be persuasive of non-obviousness in a post-*KSR* context, for example, where the Board stated, "We are persuaded by the above arguments and agree with the Appellants that one of ordinary skill would not combine Forni and Heinz in the manner suggested by the Examiner because such a combination will result in an inoperable spring pad." *In re Thomas*, Appeal 2008-0173 (April 17, 2008). Thus, Appellant submits the apparent inoperability of the purported modification of the alleged teachings of the Mann reference must be understood in the context of case law and Office practice that recognizes that evidence of "inoperability" of the purported combination of the cited prior art references undermines such an allegation of obviousness. As Appellant

alleges such "inoperability" as to the Examiner's purported combination, Appellant submits the Examiner's rejection of claims 11 and 30 is improper.

Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 11 and 30. Therefore, Appellant submits claims 11 and 30 are in condition for allowance.

Claim 12 and 31:

Regarding claims 12 and 31, Appellant submits the cited portions of the cited references do not render obvious the subject matter of claims and 31. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups to packet communications between interface groups further comprises the step of: applying interface groups to packet communications between backbone and peer interface groups." While the Examiner cites "(Figure 4, connections between peer, backbone, and customer networks at network device A)," Appellant submits the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest "...the step of: applying interface groups to packet communications between backbone and peer interface groups." Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 12 and 31. Therefore, Appellant submits claims 12 and 31 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 12 and 31, the Examiner states as follows:

> Regarding Claims 12 and 31, Applicant states that "the block diagram of Figure 4 of the Yu et al. reference does not disclose or suggest, as an example, "...the step of: applying interface groups to packet communications between backbone and peer interface groups." Examiner respectfully disagrees. Yu discloses applying interface groups to packet communications between interface groups (Figure 4, connections between peer

(e.g., between Network Device A and Network Device D) and backbone (e.g., in Network Device A between interfaces 'a' and 'd').

Appellant notes, in the Examiner's Response to Arguments with respect to claims 6 and 25, the Examiner parenthetically characterizes "the Internet" as teaching "(i.e., backbone)," without citing any reference or providing any justification or explanation as to such characterization. Now, with respect to claims 12 and 31, Appellant notes the Examiner parenthetically characterizes "(e.g., in Network Device A between interfaces 'a' and 'd')" as teaching a "backbone." Thus, Appellant submits the Examiner's assertions are inconsistent and contradictory. Appellant submits Figure 4 of the Yu reference does not disclose or suggest "applying interface groups to packet communications between backbone and peer interface groups." Appellant notes paragraph [0034] of the Yu reference recites "...define an interface group containing the tunnel interface 'd.'" However, Appellant submits Yu teaches away from "applying interface groups" to "connections between peer (e.g., between Network Device A and Network Device D) and backbone (e.g., in Network Device A between interfaces 'a' and 'd'," as Appellant submits such an alleged "applying interface groups" would appear to render inoperable the "tunnel fail over...without running a dynamic routing protocol" described in paragraph [0034] of the Yu reference. Accordingly, Appellant submits the Examiner fails to present a plausible rationale presented as to how the cited references could allegedly work together.

Appellant notes, even after *KSR International Co. v. Teleflex, Inc.*, 550 U S 398, 82 USPQ2d 1385 (2007), the Court of Appeals for the Federal Circuit has found inoperability to support a finding of non-obviousness. For example, in *DePuy Spine, Inc. v. Medtronic Sofamor Danek*, (Fed. Cir. June 1, 2009), the Court stated, "The 'predictable result' discussed in *KSR* refers not only to the expectation that prior art elements are capable of being physically combined, but also that the combination would have worked for its intended purpose," (citing *KSR*, at 1739-40) and "A reference

teaches away from a combination when using it in that combination would produce an inoperative result," (citing *In re ICON Health Fitness* (Fed. Cir. 2007). Furthermore, in *Power-One v. Artesyn* (Fed. Cir. March 30, 2010), the Court stated, "[A] patent composed of several elements is not proved obvious merely by demonstrating that each of its elements is, independently, known in the prior art." The Court could find no plausible rationale presented as to how the references could allegedly work together. Appellant submits the Board of Patent Appeals and Interferences continues to find inoperability to be persuasive of non-obviousness in a post-*KSR* context, for example, where the Board stated, "We are persuaded by the above arguments and agree with the Appellants that one of ordinary skill would not combine Forni and Heinz in the manner suggested by the Examiner because such a combination will result in an inoperable spring pad." *In re Thomas*, Appeal 2008-0173 (April 17, 2008). Thus, Appellant submits the apparent inoperability of the purported modification of the alleged teachings of the Mann reference must be understood in the context of case law and Office practice that recognizes that evidence of "inoperability" of the purported combination of the cited prior art references undermines such an allegation of obviousness. As Appellant alleges such "inoperability" as to the Examiner's purported combination, Appellant submits the Examiner's rejection of claims 12 and 31 is improper.

Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 12 and 31. Therefore, Appellant submits claims 12 and 31 are in condition for allowance.

The Fourth Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 13 and 32 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US

2002/0067725 A1) as applied to claims 4 and 23 above, and further in view of Holden et al. (United States Patent No. 5,802,178). Appellant respectfully disagrees.

Claim 13 and 32:

Regarding claims 13 and 32, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claims 13 and 32. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups further comprises the step of: applying interface groups to communication of ICMP packets." While the Examiner cites "(column 20, line 66 – column 21, line 10)," Appellant submits the cited portion of the Holden et al. reference does not disclose or suggest "...the step of: applying interface groups to communication of ICMP packets." Moreover, Appellant has presented arguments as to McDysan not disclosing the subject matter of claims from which claims 13 and 32 depend. Accordingly, even if an attempt were made to combine the teachings of the Holden reference and the McDysan reference, such an attempted combination would not yield the subject matter of claims 13 and 32. Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 13 and 32. Therefore, Appellant submits claims 13 and 32 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 13 and 32, the Examiner states as follows:

> Regarding Claims 13 and 32, Applicant states "even if an attempt were made to combine the teachings of the Holden reference and the McDysan reference, such an attempted combination would not yield the subject matter of Claims 13 and 32." Examiner respectfully disagrees. Per MPEP 2143.01: "The test for obviousness is what the combined teachings of the references would have suggested to one of ordinary skill in the art, and all teachings in the prior art must be considered to the extent that they arc in analogous arts." The McDysan, Oguchi, and Holden references are directed to processing data packets and are therefore in analogous arts. Holden further discloses a secure network interface unit (SNIU) that marks the protocol and type fields to indicate

an ICMP Echo Reply, signs the packet, and sends through an interface (column 20, line 66 - column 21, line 10).

Appellant notes the Examiner alleges "Holden further discloses a secure network interface unit (SNIU) that marks the protocol and type fields to indicate an ICMP Echo Reply." Thus, Appellant submits the Examiner appears to characterize the teachings of Holden in a manner that teaches away from the subject matter of claims 13 and 32. Appellant notes claims 13 and 32 depend indirectly from claims 1 and 20, which recite "marking packets carrying the Layer-3 control information," while the Examiner alleges teaching as to marking "an ICMP Echo Reply." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 13 and 32. Therefore, Appellant submits claims 13 and 32 are in condition for allowance.

The Fifth Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 14 and 33 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Ogushi et al. (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 4 and 23 above, and further in view of Pan et al. (United States Patent 7,336,615). Appellant respectfully disagrees.

Claim 14 and 33:

Regarding claims 14 and 33, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claims 14 and 33. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups further comprises the step of: applying interface groups to communication of ping packets." While the Examiner cites "(column 14, lines 48-55)" of the Pan reference, Appellant submits the cited portion of the Pan reference does not disclose or suggest "...the step of: applying interface

groups...." Moreover, Appellant has presented arguments as to McDysan not disclosing the subject matter of claims from which claims 14 and 33 depend. Accordingly, even if an attempt were made to combine the teachings of the Pan reference and the McDysan reference, such an attempted combination would not yield the subject matter of claims 14 and 33. Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 14 and 33. Therefore, Appellant submits claims 14 and 33 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 14 and 33, the Examiner states as follows:

> Regarding Claims 14 and 33, Applicant states "even if an attempt were made to combine the teachings of the Pan reference and the McDysan reference, such an attempted combination would not yield the subject matter of Claims 14 and 33." Examiner respectfully disagrees. Per MPEP 2143.01: "The test for obviousness is what the combined teachings of the references would have suggested to one of ordinary skill in the art, and all teachings in the prior art must be considered to the extent that they are in analogous arts." The McDysan, Oguchi, and Pan references are directed to processing data packets and are therefore in analogous arts. With regards to the claim limitation "applying interface groups to communication of ping packets," Pan discloses assigning predetermined port numbers to LSP ping messages (column 14, lines 4855).

Appellant submits "assigning predetermined port numbers to LSP ping messages" fails to disclose or suggest applying interface groups to determine when marking of control packets is to be done, wherein applying interface groups to determine when marking of control packets is to be done comprises applying interface groups to communication of ping packets, and marking packets carrying Layer-3 control information, as "assigning predetermined port numbers to LSP ping messages" does not teach or suggest "to determine when marking of control packets is to be done." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 14 and 33. Therefore, Appellant submits claims 14 and 33 are in condition for allowance.

The Sixth Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 15 and 34 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent No. 7,046,680) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 4 and 23 above, and further in view of Fotedar et al. (United States Patent Application Publication 2004/0085965 A1). Appellant respectfully disagrees.

Claim 15 and 34:

Regarding claims 15 and 34, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claims 15 and 34. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups further comprises the step of: applying interface groups to communication of traceroute packets." While the Examiner cites "(paragraph 0011)" of the Fotedar reference, Appellant submits the cited portion of the Fotedar reference does not disclose or suggest "...the step of: applying interface groups...." Moreover, Appellant has presented arguments as to McDysan not disclosing the subject matter of claims from which claims 15 and 34 depend. Accordingly, even if an attempt were made to combine the teachings of the Fotedar reference and the McDysan reference, such an attempted combination would not yield the subject matter of claims 15 and 34. Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 15 and 34. Therefore, Appellant submits claims 15 and 34 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 15 and 34, the Examiner states as follows:

> Regarding Claims 15 and 34, Applicant states "even if an attempt were made to combine the teachings of the Fotedar reference and the McDysan reference, such an attempted combination would not yield the subject matter of Claims 15 and 34." Examiner respectfully disagrees. Per MPEP 2143.01: "The test for obviousness is what the combined teachings of the references would have suggested to one of ordinary skill in the art, and all teachings in the prior art must be considered to the extent that they are in analogous arts." The McDysan, Oguchi, and Fotedar references are directed to processing data packets and are therefore in analogous arts. Further, Fotedar discloses assignment of traceroute packets to a virtual router address indicative of a loopback interface (paragraph 0011).

Appellant submits "assignment of traceroute packets to a virtual router address indicative of a loopback interface" fails to disclose or suggest applying interface groups to determine when marking of control packets is to be done, wherein applying interface groups to determine when marking of control packets is to be done comprises applying interface groups to communication of traceroute packets, and marking packets carrying Layer-3 control information, as "assignment of traceroute packets to a virtual router address indicative of a loopback interface" does not teach or suggest "to determine when marking of control packets is to be done." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 15 and 34. Therefore, Appellant submits claims 15 and 34 are in condition for allowance.

The Seventh Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 16 and 35 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 4 and 23 above, and further in view of Tuomenoksa et al. (United States Patent Application Publication 2002/0023210 A1). Appellant respectfully disagrees.

Claim 16 and 35:

Regarding claims 16 and 35, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claims 16 and 35. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest "wherein the step of applying interface groups further comprises the step of: applying interface groups to communication of packets from Network Operations Center (NOC) hosts." While the Examiner cites "(paragraph 0136)" and "(paragraphs 0141-0143)" of the Tuomenoksa reference, Appellant submits the cited portion of the Tuomenoksa reference does not disclose or suggest "...the step of: applying interface groups...." Moreover, Appellant has presented arguments as to McDysan not disclosing the subject matter of claims from which claims 16 and 35 depend. Accordingly, even if an attempt were made to combine the teachings of the Tuomenoksa reference and the McDysan reference, such an attempted combination would not yield the subject matter of claims 16 and 35. Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 16 and 35. Therefore, Appellant submits claims 16 and 35 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 16 and 35, the Examiner states as follows:

> Regarding Claims 16 and 35, Applicant states "even if an attempt were made to combine the teachings of the Tuomenoksa reference and the McDysan reference, such an attempted combination would not yield the subject matter of Claims 16 and 35." Examiner respectfully disagrees. Per MPEP 2143.01: "The test for obviousness is what the combined teachings of the references would have suggested to one of ordinary skill in the art, and all teachings in the prior art must be considered to the extent that they are in analogous arts." The McDysan, Oguchi, and Tuomenoksa references are directed to processing data packets and are therefore in analogous arts. Further, Tuomenoksa discloses setting up a tunnel interface with a NOC (paragraph 0136) and communicating packets, including control information, with the NOC via the tunnel (paragraphs 0141-0143).

Appellant submits "setting up a tunnel interface with a NOC (paragraph 0136) and communicating packets, including control information, with the NOC via the tunnel (paragraphs 0141-0143)" fails to disclose or suggest applying interface groups to determine when marking of control packets is to be done, wherein applying interface groups to determine when marking of control packets is to be done comprises applying interface groups to communication of packets from Network Operations Center (NOC) hosts, and marking packets carrying Layer-3 control information, as "setting up a tunnel interface with a NOC (paragraph 0136) and communicating packets, including control information, with the NOC via the tunnel (paragraphs 0141-0143)" does not teach or suggest "to determine when marking of control packets is to be done." Moreover, Appellant submits "setting up a tunnel interface with a NOC" does not disclose or suggest "applying interface groups to communication of packets from Network Operations Center (NOC) hosts." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 16 and 35. Therefore, Appellant submits claims 16 and 35 are in condition for allowance.

The Eighth Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 18 and 37 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 1 and 20 above, and further in view of Johansson (United States Patent 6,061,330). Appellant respectfully disagrees.

Claim 18 and 37:

Regarding claims 18 and 37, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claims 18 and 37. As an example,

Appellant submits the cited portions of the cited references do not disclose or suggest "receiving unmarked control packets using rate-limited queues." While the Examiner cites "(Figure 1, 116; Figure 4a, 410)" of the Johansson reference, Appellant submits the Examiner does not allege the cited portion of the Johansson reference as disclosing or suggesting "receiving unmarked control packets using rate-limited queues." Rather, Appellant notes the Examiner alleges McDysan discloses "the unmarked control packets (i.e., packets received prior to being marked)" without citing any portion of the McDysan reference as allegedly teaching such subject matter. Moreover, Appellant has presented arguments as to McDysan not disclosing the subject matter of claims from which claims 18 and 37 depend. For example, Appellant submits McDysan fails to disclose "...control packets." Accordingly, even if an attempt were made to combine the teachings of the Johansson reference and the McDysan reference, such an attempted combination would not yield the subject matter of claims 18 and 37. Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 18 and 37. Therefore, Appellant submits claims 18 and 37 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 18 and 37, the Examiner states as follows:

> Regarding Claims 18 and 37, Applicant states that the cited portions of the cited references do not disclose "control packets." Applicant further states "even if an attempt were made to combine the teachings of the Johansson reference and the McDysan reference, such an attempted combination would not yield the subject matter of Claims 18 and 37." Examiner respectfully disagrees. The claim language "control packets" is not further defined in the claim language so as to further limit the content or structure of the claimed "control packet." As such, Examiner has given the claim term its broadest reasonable interpretation without unnecessarily importing limitations from the specification and interpreted "control packet" to comprise any messaging related to control of communications (e.g., setup, teardown, parameter management, etc.). Further, per MPEP 2143.01: "The test for obviousness is what the combined teachings of the references would have suggested to one of ordinary skill in the art, and all teachings in the prior art must be considered to the extent that they are in analogous arts." The McDysan, Oguchi, and Johansson references are directed to processing data

> packets and are therefore in analogous arts." While McDysan discloses processing control information in a network (paragraphs 0037 and 0042), the combination of McDysan and Oguchi does not disclose processing the control packets at a line rate. In the same field of endeavor, Figure 4a, step 410 of Johansson "determines when a predetermined number Input RateLimit of Cells are received" (column 10, lines 45-47). As such, Johansson provides a general teaching of a rate-limited queue receiving packets.

Appellant submits the Examiner does not appear to allege any teaching or suggestion as to, for example, "unmarked control packets." Rather, Appellant notes, with respect to claims 1 and 20, from which claims 18 and 37 depend, the Examiner alleges "...McDysan discloses marking packets via a DSCP code point in IP packet...." Thus, Appellant submits the combination of references cited by the Examiner appear to teach away from "unmarked control packets." Moreover, Appellant submits the "cells" of Johansson fail to disclose or suggest "unmarked control packets." Accordingly, Appellant submits the Examiner has not made a *prima facie* showing of obviousness as to claims 18 and 37 Therefore, Appellant submits claims 18 and 37 are in condition for allowance.

The Ninth Ground of Rejection to be Reviewed upon Appeal:

The Examiner has rejected claims 19 and 38 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McDysan et al. (U.S. Patent Application Publication 2003/0112755 A1) in view of Oguchi et al. (U.S. Patent Publication No. US 2002/0067725 A1) as applied to claims 1 and 20 above, and further in view of Hussey et al. (United States Patent Application Publication 2001/0049744 A1). Appellant respectfully disagrees.

Claim 19 and 38:

Regarding claims 19 and 38, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claims 19 and 38. As an example,

Appellant submits the cited portions of the cited references do not disclose or suggest "receiving the packets as received packets; and processing the received packets at a line rate." While the Examiner cites "(paragraph 0050)" of the Hussey reference, Appellant submits "(paragraph 0050)" of the Hussey reference states, in part, "...receives a packet data stream via the communication network 110 at a line rate...." Appellant submits such teaching does not disclose or suggest "receiving the packets as received packets" and "processing the received packets at a line rate." Moreover, Appellant has presented arguments as to McDysan not disclosing the subject matter of claims from which claims 19 and 38 depend. Accordingly, even if an attempt were made to combine the teachings of the Hussey reference and the McDysan reference, such an attempted combination would not yield the subject matter of claims 19 and 38. Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 19 and 38. Therefore, Appellant submits claims 19 and 38 are in condition for allowance.

In the Examiner's Response to Arguments, regarding claims 19 and 38, the Examiner states as follows:

> Regarding Claims 19 and 38, Applicant states "even if an attempt were made to combine the teachings of the Johansson reference and the McDysan reference, such an attempted combination would not yield the subject matter of Claims 19 and 38." Examiner notes that the Johanson reference is not relied upon for rejection of Claims 19 and 38 under 35 U.S.C. 103(a). Per MPEP 2143.01: "The test for obviousness is what the combined teachings of the references would have suggested to one of ordinary skill in the art, and all teachings in the prior art must be considered to the extent that they are in analogous arts." The McDysan, Oguchi, and Hussey references are directed to processing data packets and are therefore in analogous arts. Further, Hussey discloses a processor pool aggregation technique wherein a communication device "receives a packet data stream via the communication network ... at a line rate that might otherwise overwhelm the processing capabilities of the NIC ... and result in dropped packets and reduced quality of service" (paragraph 0050).

Regarding claims 19 and 38, Appellant submits the cited portions of the cited references fail to render obvious the subject matter of claims 19 and 38. As an example, Appellant submits the cited portions of the cited references do not disclose or suggest

"receiving the packets as received packets; and processing the received packets at a line rate." While the Examiner cites "(paragraph 0050)" of the Hussey reference, Appellant submits "(paragraph 0050)" of the Hussey reference states, in part, "...receives a packet data stream via the communication network 110 at a line rate...." Appellant submits such teaching does not disclose or suggest "receiving the packets as received packets" and "processing the received packets at a line rate." Moreover, Appellant has presented arguments as to McDysan not disclosing the subject matter of claims from which claims 19 and 38 depend. Accordingly, even if an attempt were made to combine the teachings of the Hussey reference and the McDysan reference, such an attempted combination would not yield the subject matter of claims 19 and 38. Thus, Appellant submits the Examiner has not made a *prima facie* showing of obviousness with respect to claims 19 and 38. Therefore, Appellant submits claims 19 and 38 are in condition for allowance.

CLAIMS APPENDIX

1. (Previously Presented) A method for communicating Layer-3 control information in a communications network comprising the steps of:
marking packets carrying the Layer-3 control information;
encapsulating the packets at Layer-2 to uniquely identify Layer-2 frames as carrying trusted control information.

2. (Original) The method of claim 1 wherein the step of marking further comprises:
marking the packets using a unique protocol identifier.

3. (Original) The method of claim 1 wherein the step of marking further comprises:
marking the packets using a link-local MPLS label.

4. (Original) The method of claim 1 further comprising the step of:
applying interface groups to determine when marking of control packets is to be done.

5. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:
applying interface groups to packet communications within a particular interface group.

6. (Original) The method of claim 5 wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of:

applying interface groups to packet communications within a backbone interface group.

7. (Original) The method of claim 5 wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of:

applying interface groups to packet communications within a customer-specific interface group.

8. (Original) The method of claim 5 wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of:

applying interface groups to packet communications within a peer interface group.

9. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:

applying interface groups to packet communications between interface groups.

10. (Original) The method of claim 9 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between backbone and customer-specific interface groups.

11. (Original) The method of claim 9 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between customer-specific and peer interface groups.

12. (Original) The method of claim 9 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between backbone and peer interface groups.

13. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of ICMP packets.

14. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of ping packets.

15. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of traceroute packets.

16. (Original) The method of claim 4 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of packets from Network Operations Center (NOC) hosts.

17. (Original) The method of claim 1 wherein the step of encapsulating the packets further comprises:
encapsulating the packets according to control encapsulation.

18. (Original) The method of claim 1 further comprising:
receiving unmarked control packets using rate-limited queues.

19. (Original) The method of claim 1 further comprising:
receiving the packets as received packets; and
processing the received packets at a line rate.

20. (Previously Presented) An apparatus comprising a network element for communicating Layer-3 control information in a communications network adapted to perform the steps of:
marking packets carrying the Layer-3 control information;
encapsulating the packets at Layer-2 to uniquely identify Layer-2 frames as carrying trusted control information.

21. (Original) The apparatus of claim 20 wherein the step of marking further comprises:
marking the packets using a unique protocol identifier.

22. (Original) The apparatus of claim 20 wherein the step of marking further comprises:
marking the packets using a link-local MPLS label.

23. (Original) The apparatus of claim 20 wherein the network element is further adapted to perform the step of:
applying interface groups to determine when marking of control packets is to be done.

24. (Original) The apparatus of claim 23 wherein the step of applying interface groups further comprises the step of:
applying interface groups to packet communications within a particular interface group.

25. (Original) The apparatus of claim 24 wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of:
applying interface groups to packet communications within a backbone interface group.

26. (Original) The apparatus of claim 24 wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of:
applying interface groups to packet communications within a customer-specific interface group.

27. (Original) The apparatus of claim 24 wherein the step of applying interface groups to packet communications within a particular interface group further comprises the step of:
applying interface groups to packet communications within a peer interface group.

28. (Original) The apparatus of claim 23 wherein the step of applying interface groups further comprises the step of:

applying interface groups to packet communications between interface groups.

29. (Original) The apparatus of claim 28 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between backbone and customer-specific interface groups.

30. (Original) The apparatus of claim 28 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between customer-specific and peer interface groups.

31. (Original) The apparatus of claim 28 wherein the step of applying interface groups to packet communications between interface groups further comprises the step of:

applying interface groups to packet communications between backbone and peer interface groups.

32. (Original) The apparatus of claim 23 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of ICMP packets.

33. (Original) The apparatus of claim 23 wherein the step of applying interface groups further comprises the step of:

applying interface groups to communication of ping packets.

34. (Original) The apparatus of claim 23 wherein the step of applying interface groups further comprises the step of:
applying interface groups to communication of traceroute packets.

35. (Original) The apparatus of claim 23 wherein the step of applying interface groups further comprises the step of:
applying interface groups to communication of packets from Network Operations Center (NOC) hosts.

36. (Original) The apparatus of claim 20 wherein network element is further adapted to encapsulate the packets according to control encapsulation.

37. (Original) The method of claim 20 wherein the network element is further adapted to receive unmarked control packets using rate-limited queues.

38. (Original) The apparatus of claim 20 wherein the network element is further adapted to receive the packets as received packets and to process the received packets at a line rate.

EVIDENCE APPENDIX

As presently advised, no evidence was submitted pursuant to 37 C.F.R. §§ 1.130, 1.131, or 1.132. Appellant mailed an information disclosure statement PTO/SB/08 on June 7, 2004, citing VIJAY GILL, JOHN HEASLY, and DAVID MEYER; The BGP TTL Security Hack (BTSH); 2002; Phoenix, AZ, USA. Appellant mailed an information disclosure statement PTO/SB/08 on November 15, 2004, citing JOHANSSON, P.; "IPv4 Over IEEE 1394"; Network Working Group Request for Comments: 2734; December 1999; Pgs. 1-29; The Internet Society; H.T. KUNG, ET AL.; "TCP Trunking: Design, Implementation and Performance"; October 31,1999; Pgs. 222-231; HARVARD UNIVERSITY; Cambridge, MA, USA; ARMITAGE, GRENVILLE; "MPLS: The Magic Behind the Myths"; IEEE Communications Magazine; January 2000; Pgs. 124-131; IEEE Service Center; Piscataway, N.J.; USA; M. BALTATU, ET AL.; "Security Issues in Control, Management and Routing Protocols"; Computer Networks; December 2000; Pgs. 881-894; Volume 34, No. 6; Elsevier Science Publishers B.V.; Amsterdam,, NL; and ED OSKIEWICZ, ET AL.; "A Model for Interface Groups"; ANSA; May 19, 1994; Pgs. 1-38; Retrieved From the Internet; Cambridge, United Kingdom. Appellant mailed an information disclosure statement PTO/SB/08 on February 15, 2005, citing WO 01/69852 A2, 09-20-2001, of Riverdelta Networks, and HLUCHYJ M G ET AL.: "Queueing Disciplines for Integrated Fast Packet Networks" Discovering a New World of Communications, Chicago, June 14-18, 1992, Pages 990-996, Vol. 4, IEEE, New York, USA. In the Notice of References Cited (Form PTO-892) included with the Office action mailed June 27, 2007, the Examiner cited U.S. Patent No. 6,731,652, issued to Ramfelt et al., U.S. Patent No. 7,126,952, issued to Hooper et al., and U.S. Patent Application Publication 2003/0112749, of Hassink et al. In the Notice of References Cited (Form PTO-892) included with the Office action mailed April 23, 2008, the Examiner cited U.S. Patent No. 7,046,680, issued to McDysan et al., U.S. Patent Application

Publication 2002/0116501 A1, of Ho et al., U.S. Patent No. 6,061,330, issued to Johansson, U.S. Patent Application Publication 2001/0049744 A1, of Hussey et al., U.S. Patent Application Publication 2004/0054924 A1, of Chuah et al., U.S. Patent Application Publication 2004/0010583 A1, of Yu et al., U.S. Patent 7,336,615, issued to Pan et al., U.S. Patent Application Publication 2002/0085498 A1, of Nakamichi et al., and U.S. Patent Application Publication 2004/0085965 A1, of Fotedar. With Appellant's response mailed July 23, 2008, Appellant included a copy of the then-current Wikipedia (http://en.wikipedia.org) entry for Layer 2 Tunneling Protocol (L2TP). In the Notice of References Cited (Form PTO-892) included with the Office action mailed November 24, 2008, the Examiner additionally cited U.S. Patent Application Publication 2003/0152078 A1, of Henderson et al. In the Notice of References Cited (Form PTO-892) included with the Office action mailed June 5, 2009, the Examiner additionally cited Almquist, RFC 1349: Type of Service in the Internet Protocol Suite, July 1992, Internet Engineering Task Force Network Working Group and The OSI Reference Model, October 20, 2002, Available online: http://web.archive.org/web/20021020084747/http://www.thecertificationhub.com/networkplus/the_osi_ref_model.htm. In the Notice of References Cited (Form PTO-892) included with the Office action mailed February 22, 2010, the Examiner cited U.S. Patent No. 5,802,178, issued to Holden et al.; U.S. Patent Application Publication 2002/0023210, of Tuomenoksa et al.; U.S. Patent Application Publication 2003/0112755, of McDysan; and U.S. Patent Application Publication 2002/0067725, of Oguchi et al. As the Examiner relied upon the following evidence in support of the final rejection, Appellant relies upon such evidence in the appeal: U.S. Patent Application Publication 2003/0112755, of McDysan; U.S. Patent Application Publication 2002/0067725, of Oguchi et al.; U.S. Patent Application Publication 2002/0085498 A1, of Nakamichi et al.; U.S. Patent Application Publication 2004/0010583 A1, of Yu et al.; U.S. Patent No. 5,802,178, issued to Holden et al.; U.S. Patent 7,336,615, issued to Pan

et al.; U.S. Patent Application Publication 2004/0085965 A1, of Fotedar; U.S. Patent Application Publication 2002/0023210, of Tuomenoksa et al.; U.S. Patent No. 6,061,330, issued to Johansson; and U.S. Patent Application Publication 2001/0049744 A1, of Hussey et al., copies of which are provided below.

US 20030112755A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2003/0112755 A1

McDysan (43) Pub. Date: Jun. 19, 2003

(54) VIRTUAL PRIVATE NETWORK (VPN)-AWARE CUSTOMER PREMISES EQUIPMENT (CPE) EDGE ROUTER

(75) Inventor: David E. McDysan, Herndon, VA (US)

Correspondence Address:
WORLDCOM, INC.
TECHNOLOGY LAW DEPARTMENT
1133 19TH STREET NW
WASHINGTON, DC 20036 (US)

(73) Assignee: WorldCom, Inc., Jackson, MS (US)

(21) Appl. No.: 10/023,331

(22) Filed: Dec. 17, 2001

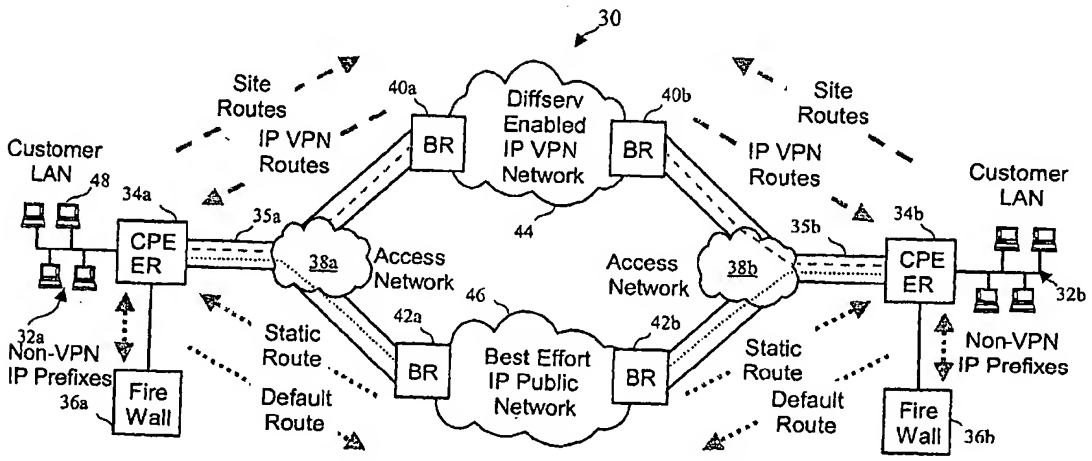
Publication Classification

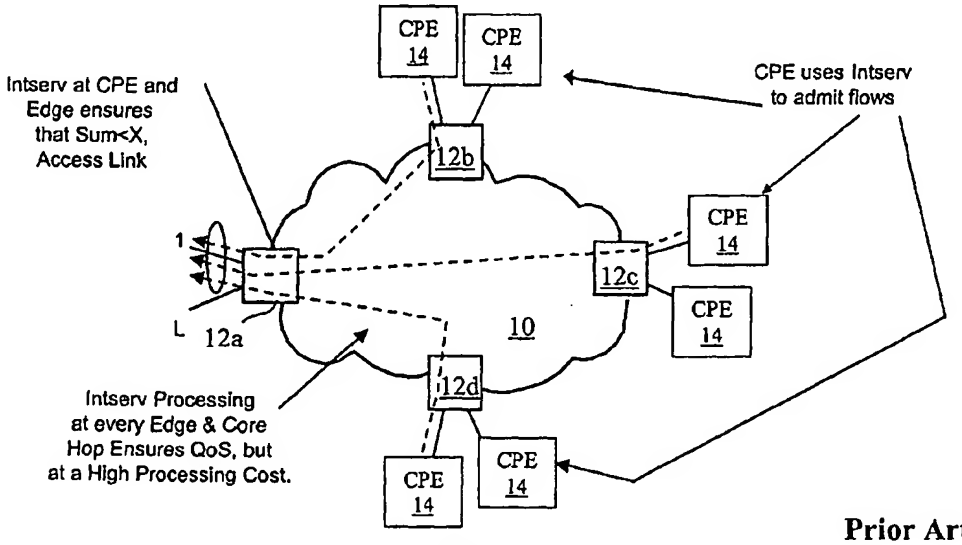
(51) Int. Cl.[7] H04J 3/14; H04L 12/56

(52) U.S. Cl. 370/230; 370/401

(57) ABSTRACT

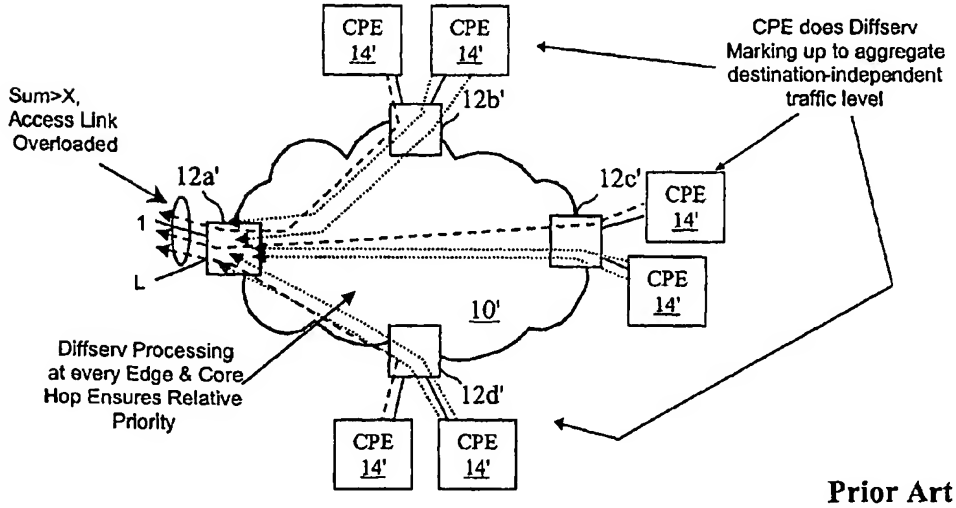
A network architecture includes a communication network that supports one or more network-based Virtual Private Networks (VPNs). The communication network includes a plurality of boundary routers that are connected by access links to CPE edge routers belonging to the one or more VPNs. To prevent traffic from outside a customer's VPN (e.g., traffic from other VPNs or the Internet at large) from degrading the QoS provided to traffic from within the customer's VPN, the present invention gives precedence to intra-VPN traffic over extra-VPN traffic on each customer's access link through access link prioritization or access link capacity allocation, such that extra-VPN traffic cannot interfere with inter-VPN traffic. Granting precedence to intra-VPN traffic over extra-VPN traffic in this manner entails partitioning between intra-VPN and extra-VPN traffic on the physical access link using layer 2 multiplexing and configuration of routing protocols to achieve logical traffic separation between intra-VPN traffic and extra-VPN traffic at the VPN boundary routers and CPE edge routers. By configuring the access networks, the VPN boundary routers and CPE edge routers, and the routing protocols of the edge and boundary routers in this manner, the high-level service of DoS attack prevention is achieved.





Prior Art

Figure 1



Prior Art

Figure 2

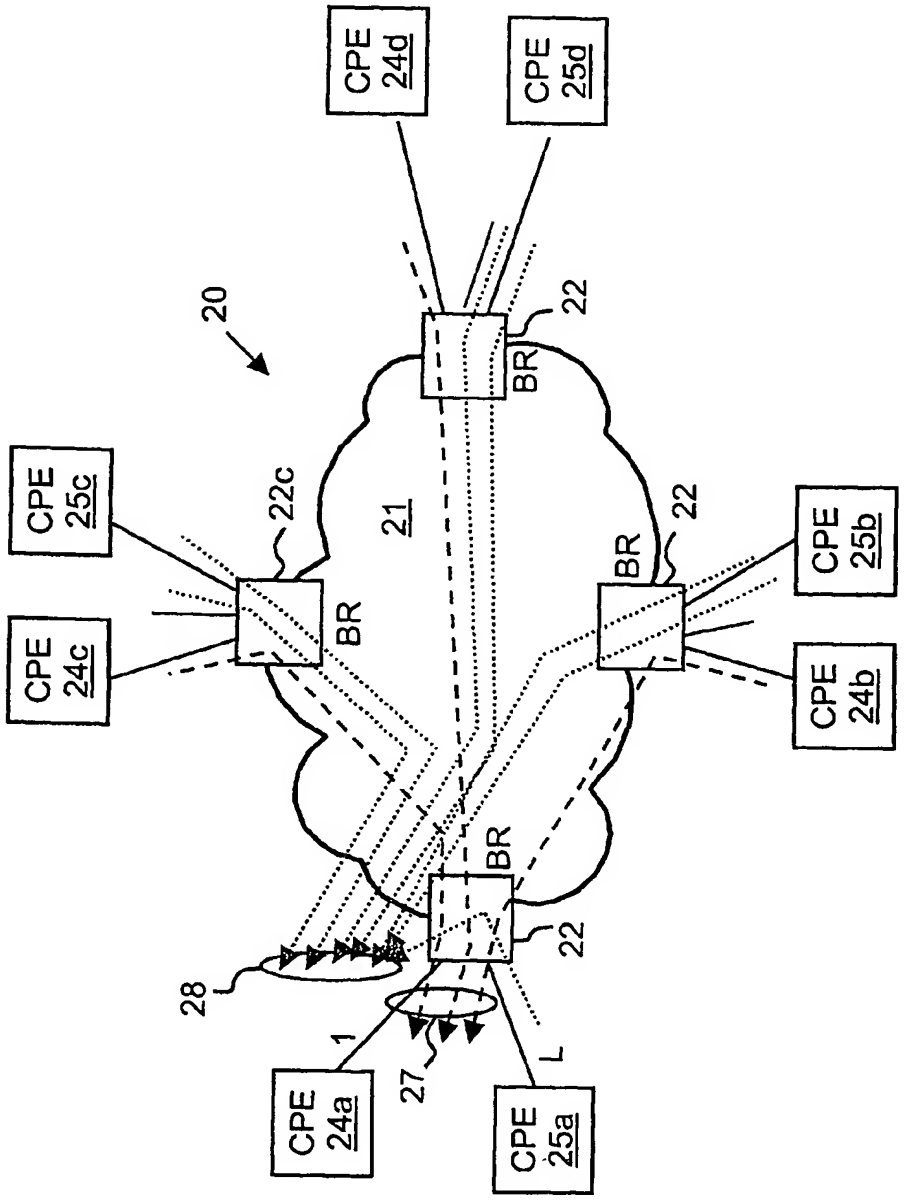

CPE 24d
CPE 25d
20
22
BR
CPE 25c
22c
21
22
BR
CPE 25b
CPE 24c
BR
CPE 24b
BR
22
28
1
27
L
CPE 24a
CPE 25a

Figure 3

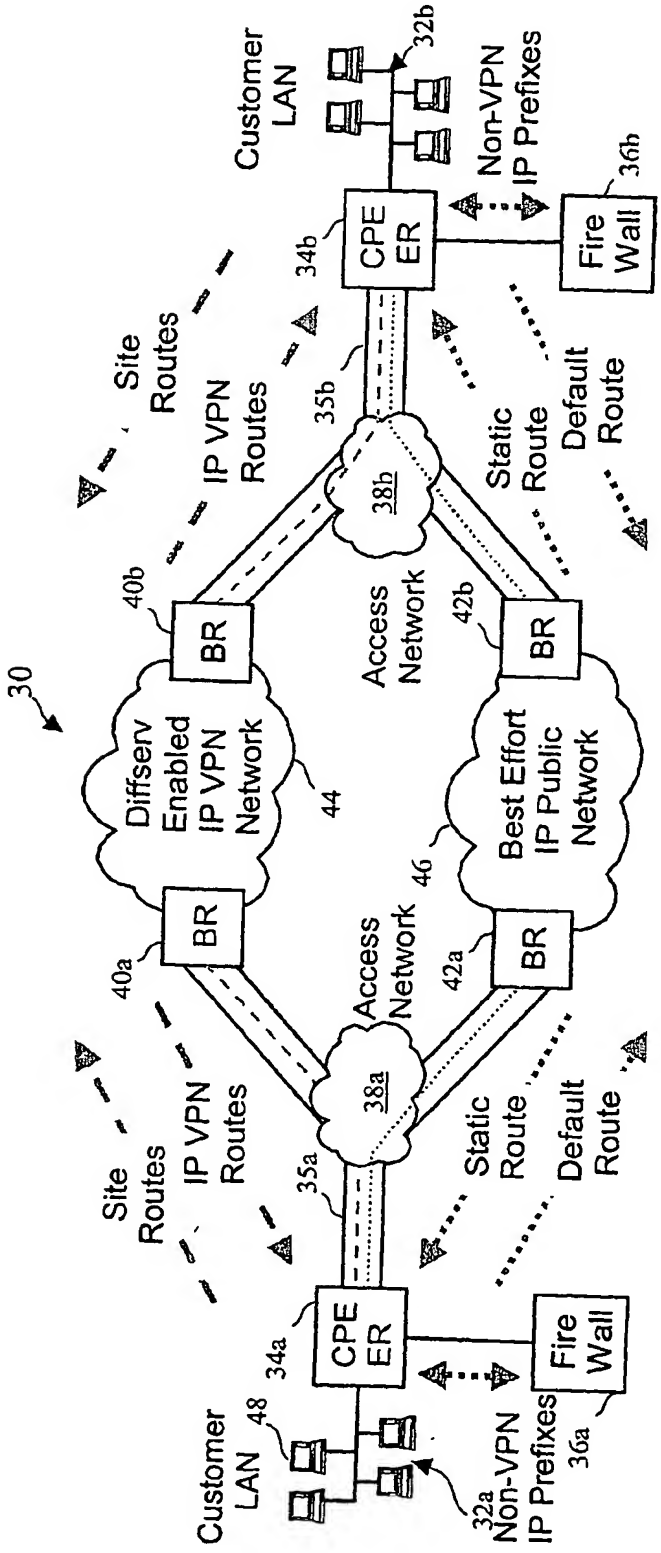

30
Customer LAN
32b
Non-VPN IP Prefixes
36b
Fire Wall
34b
CPE ER
Site Routes
IP VPN Routes
35b
38b
Static Route
Default Route
40b
BR
Access Network
42b
BR
Diffserv Enabled IP VPN Network
44
Best Effort IP Public Network
46
40a
BR
Access Network
42a
BR
Site Routes
IP VPN Routes
35a
38a
Static Route
Default Route
34a
CPE ER
Fire Wall
36a
Customer LAN
48
32a
Non-VPN IP Prefixes

Figure 4

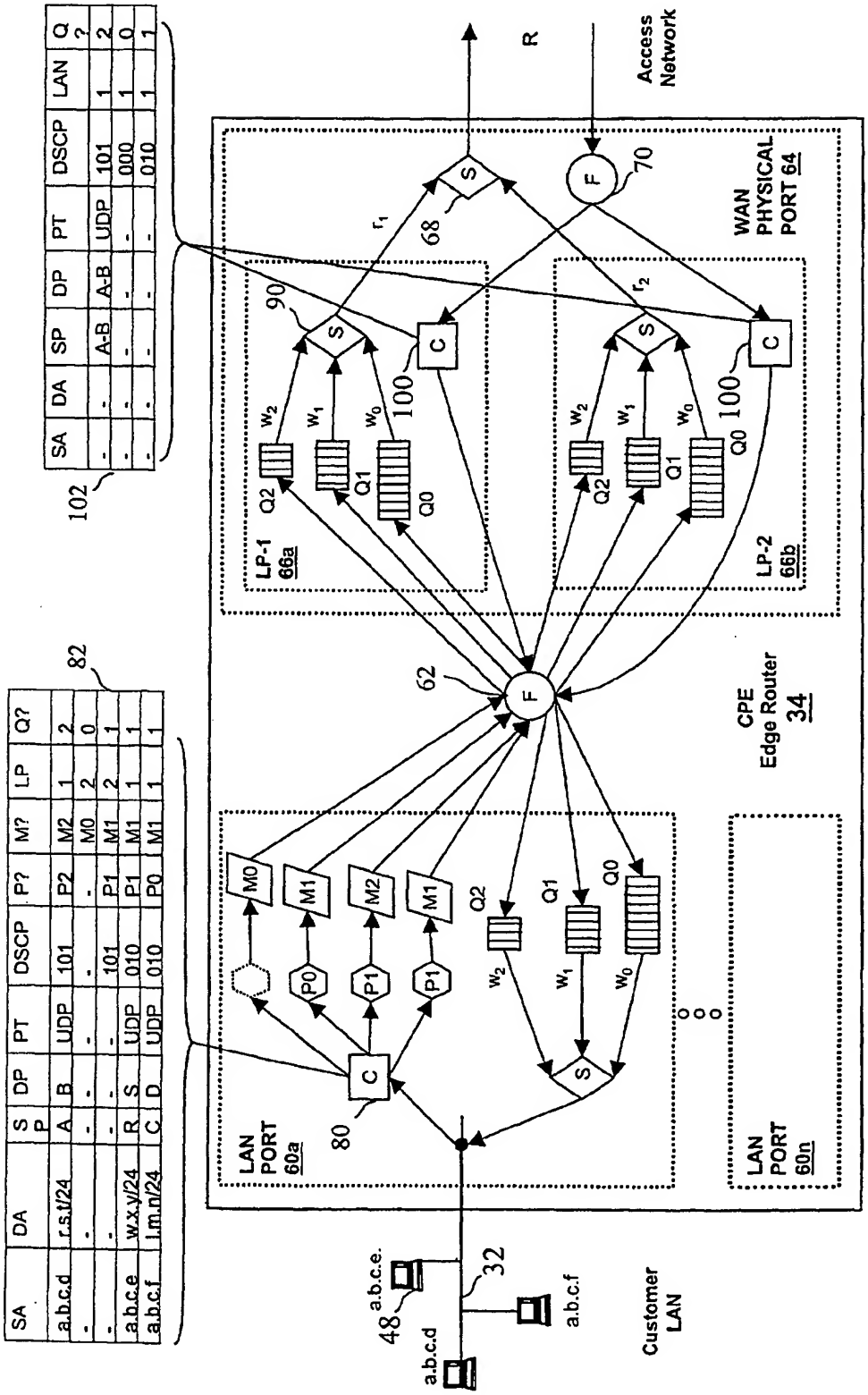

SA DA SP DP PT DSCP LAN Q?
- - - - - - 1 2
- - A-B - - 000 1 0
- - - - - 010 1 1
102
SA DA SP DP PT DSCP P? M? LP Q?
a.b.c.d r.s.t/24 A B UDP 101 P2 M2 1 2
- - - - - - P0 M0 2 0
- - - - - 101 P1 M1 2 1
a.b.c.e w.x.y/24 R S UDP 010 P1 M1 1 1
a.b.c.f l.m.n/24 C D UDP 010 P0 M1 1 1
82
Access Network
R
F
70
S
68
r_1
r_2
WAN PHYSICAL PORT 64
LP-1 66a
LP-2 66b
90
100
w_2
w_1
w_0
Q2
Q1
Q0
C
62
CPE Edge Router 34
LAN PORT 60a
LAN PORT 60n
80
M0
M1
M2
P0
P1
32
48
a.b.c.e
a.b.c.d
a.b.c.f
Customer LAN

Figure 5

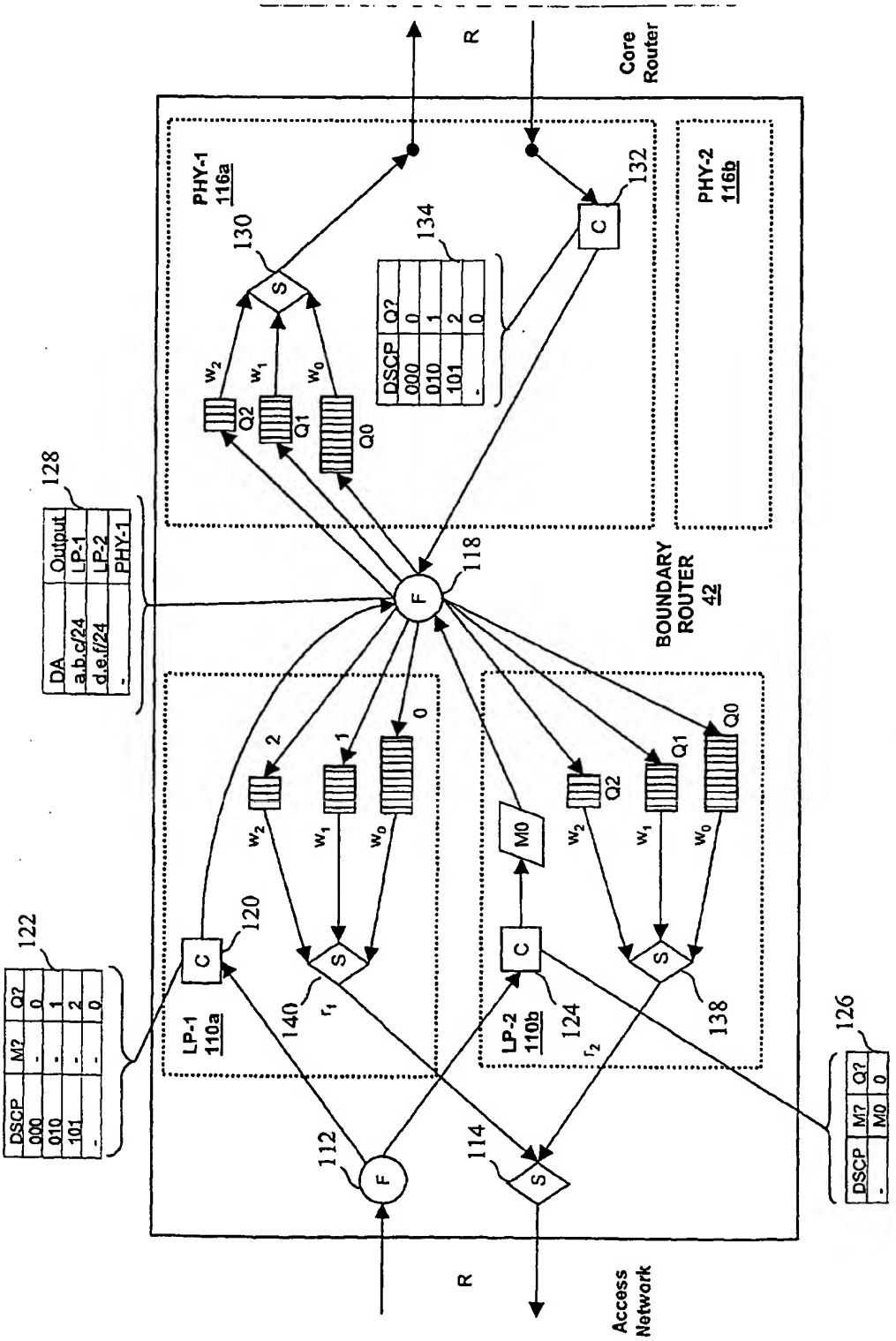

R
Core Router
PHY-1 116a
PHY-2 116b
130
132
134
S
C
DSCP Q2
000 0
010 1
101 2
- 0
w2
w1
w0
Q2
Q1
Q0
128
DA Output
a.b.c/24 LP-1
d.e.f/24 LP-2
- PHY-1
118
F
BOUNDARY ROUTER 42
2
1
0
M0
Q2
Q1
Q0
122
DSCP M? Q?
000 - 0
010 - 1
101 - 2
- - 0
120
LP-1 110a
LP-2 110b
140
r1
r2
124
138
126
DSCP M? Q?
- M0 0
112
114
R
Access Network

Figure 6A

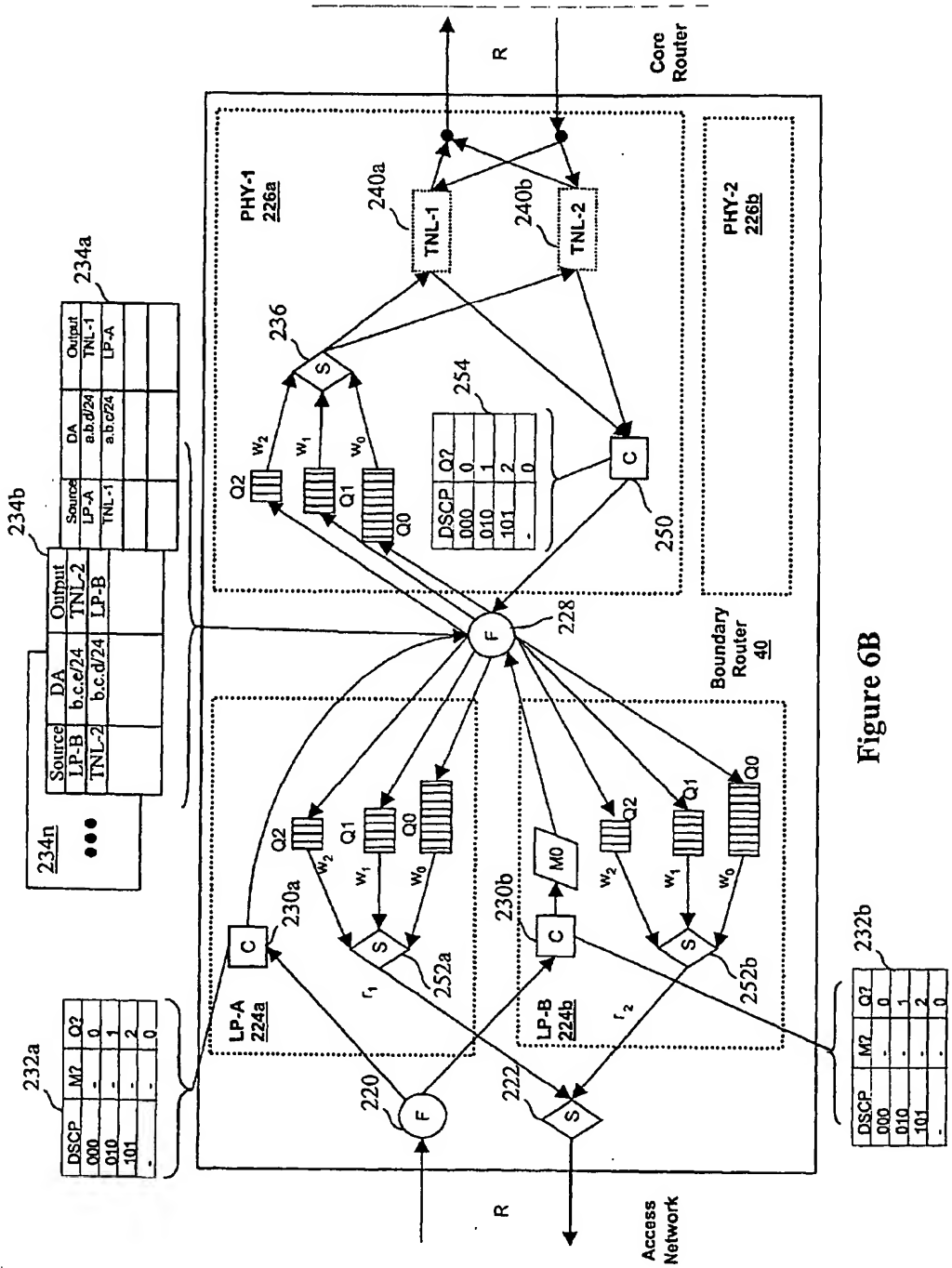

Core Router
R
PHY-1 226a
PHY-2 226b
240a
TNL-1
240b
TNL-2
236
S
254
DSCP Q2
000 0
010 1
101 2
- 0
250
C
Q2
Q1
Q0
w_2
w_1
w_0
228
F
Boundary Router 40
234a
Source DA Output
LP-A a.b.d/24 TNL-1
TNL-1 a.b.c/24 LP-A
234b
Source DA Output
LP-B b.c.e/24 TNL-2
TNL-2 b.c.d/24 LP-B
234n
230a
230b
M0
LP-A 224a
LP-B 224b
r_1
r_2
252a
252b
220
222
232a
DSCP M2 Q2
000 - 0
010 - 1
101 - 2
- - 0
232b
DSCP M2 Q2
000 - 0
010 - 1
101 - 2
- - 0
R
Access Network

Figure 6B

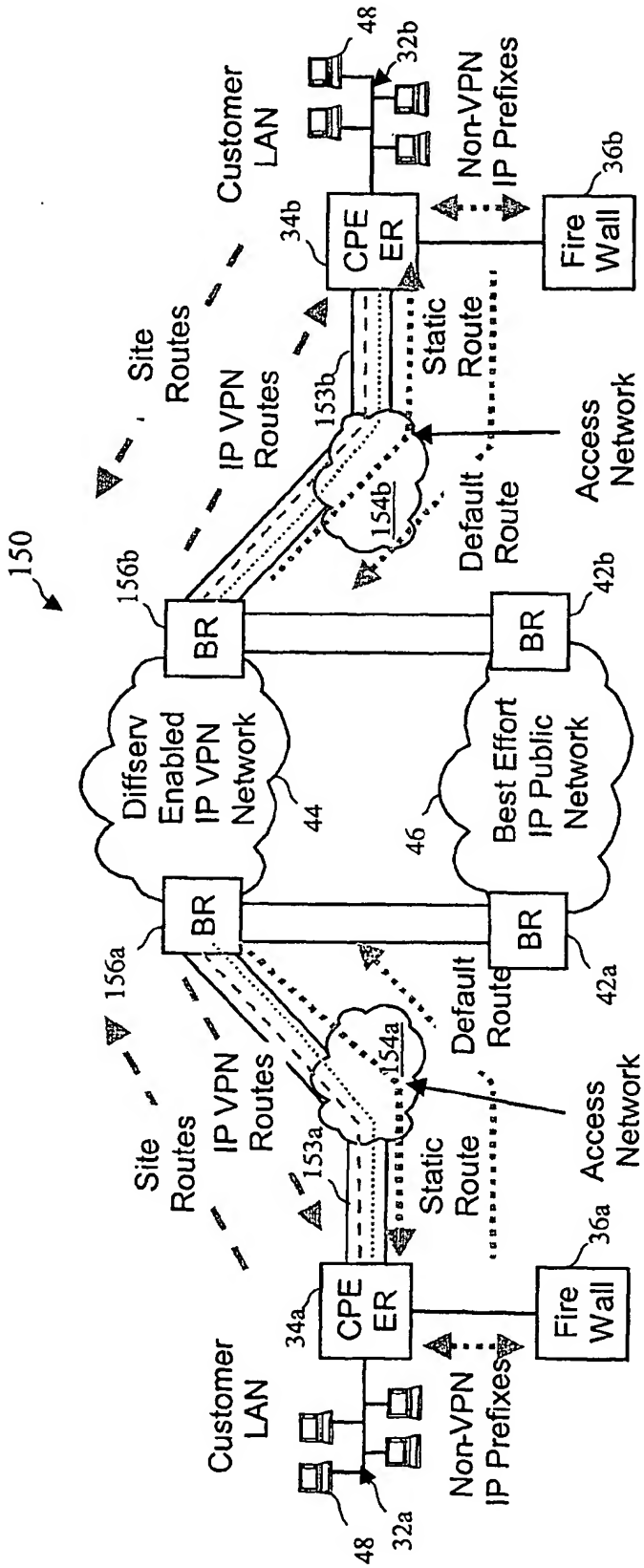

150
Diffserv Enabled IP VPN Network
44
Best Effort IP Public Network
46
BR
156a
156b
42a
42b
Site Routes
IP VPN Routes
Customer LAN
Non-VPN IP Prefixes
Static Route
Default Route
Access Network
CPE ER
34a
34b
Fire Wall
36a
36b
153a
153b
154a
154b
32a
32b
48

Figure 7

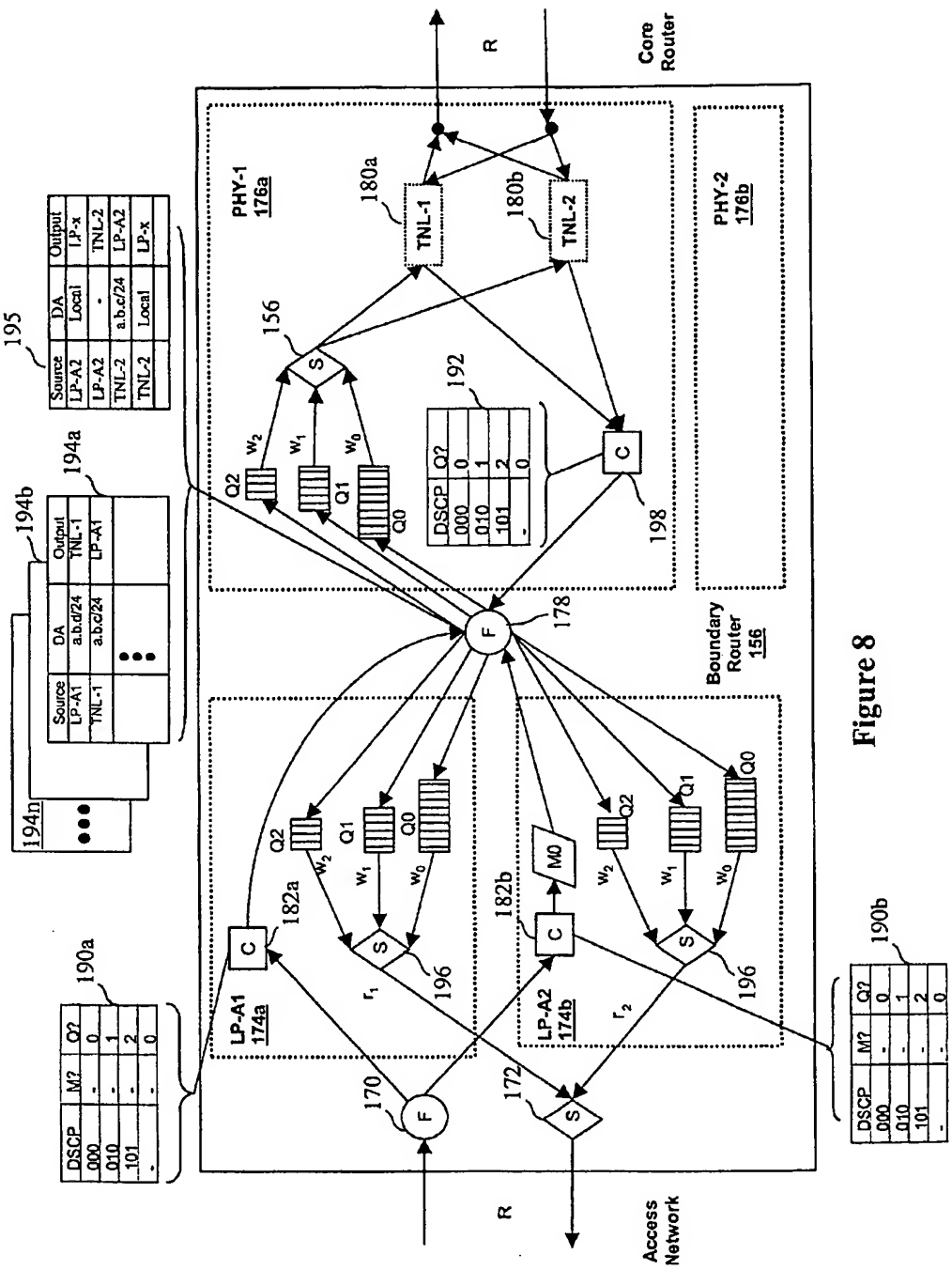

Core Router
R
PHY-1 176a
PHY-2 176b
180a
TNL-1
180b
TNL-2
156
S
192
DSCP Q2
000 0
010 1
101 2
- 0
C
198
Q2
Q1
Q0
w2
w1
w0
178
F
Boundary Router 156
195
Source DA Output
LP-A2 Local LP-x
LP-A2 - TNL-2
TNL-2 a.b.c/24 LP-A2
TNL-2 Local LP-x
194a
194b
194n
Source DA Output
LP-A1 a.b.d/24 TNL-1
TNL-1 a.b.c/24 LP-A1
182a
182b
M0
LP-A1 174a
LP-A2 174b
196
r1
r2
170
172
190a
DSCP M? Q?
000 - 0
010 - 1
101 - 2
- - 0
190b
DSCP M? Q?
000 - 0
010 - 1
101 - 2
- - 0
R
Access Network

Figure 8

VIRTUAL PRIVATE NETWORK (VPN)-AWARE CUSTOMER PREMISES EQUIPMENT (CPE) EDGE ROUTER

BACKGROUND OF THE INVENTION

[0001] 1. Technical Field

[0002] The present invention relates to communication networks and, in particular, to the prevention of denial of service attacks in a public communication network, for example, the Internet. Still more particularly, the present invention relates to method, system and apparatus for preventing denial of service attacks in a communication network having a shared network infrastructure by separating the allocation and/or prioritization of access capacity to traffic of sites within a virtual private network (VPN) from the allocation and/or prioritization of access capacity to sites in another VPN or the public network.

[0003] 2. Description of the Related Art

[0004] For network service providers, a key consideration in network design and management is the appropriate allocation of access capacity and network resources between traffic originating from VPN customer sites and traffic originating from outside the VPN (e.g., from the Internet or other VPNs). This consideration is particularly significant with respect to the traffic of VPN customers whose subscription includes a Service Level Agreement (SLA) requiring the network service provider to provide a minimum communication bandwidth or to guarantee a particular Quality of Service (QoS). Such service offerings require the network service provider to implement a network architecture and protocol that achieve a specified QoS and ensure sufficient access capacity and network resources are available for communication with other VPN sites separate from communication with hosts that are not part of the VPN.

[0005] In Internet Protocol (IP) networks, a straightforward approach to achieving QoS and implementing admission control comparable to that of connection-oriented network services, such as voice or Asynchronous Transfer Mode (ATM), is to emulate the same hop-by-hop switching paradigm of signaling resource reservations for the flow of IP packets requiring QoS.

[0006] In fact, the IP signaling standard developed by the Internet Engineering Task Force (IETF) for Integrated Services (Intserv) adopts precisely this approach. As described in IETF RFC 1633, Intserv is a per-flow IP QoS architecture that enables applications to choose among multiple, controlled levels of delivery service for their data packets. To support this capability, Intserv permits an application at a transmitter of a packet flow to use the well-known Resource ReSerVation Protocol (RSVP) defined by IETF RFC 2205 to request a desired QoS class at a specific level of capacity from all network elements along the path to a receiver of the packet flow. After receiving an RSVP PATH message requesting a resource reservation and an RSVP RESV message confirming resource reservation from an upstream node, individual network elements along the path implement mechanisms to control the QoS and capacity delivered to packets within the flow.

[0007] **FIG. 1** illustrates the implications of utilizing a conventional Intserv implementation to perform admission control. As shown in **FIG. 1**, an exemplary IP network **10** includes N identical nodes (e.g., service provider boundary routers) **12**, each having L links of capacity X coupled to Customer Premises Equipment (CPE) **14** for L distinct customers. In a per-flow, connection-oriented approach, each node **12** ensures that no link along a network path from source to destination is overloaded. Looking at access capacity, a per-flow approach is able to straightforwardly limit the input flows on each of the ingress access links such that the sum of the capacity for all flows does not exceed the capacity X of any egress access link (e.g., Link **1** of node **12***a*). A similar approach is applicable to links connecting unillustrated core routers within IP network **10**.

[0008] Although conceptually very simple, the admission control technique illustrated in **FIG. 1** has a number of drawbacks. Most importantly, Intserv admission control utilizing RSVP has limited scalability because of the processing-intensive signaling RSVP requires in the service provider's boundary and core routers. In particular, RSVP requires end-to-end signaling to request appropriate resource allocation at each network element between the transmitter and receiver, policy queries by ingress node **12***b*-**12***d* to determine which flows to admit and police their traffic accordingly, as well as numerous other handshake messages. Consequently, the processing required by Intserv RSVP signaling is comparable to that of telephone or ATM signaling and requires a high performance (i.e., expensive) processor component within each boundary or core IP router to handle the extensive processing required by such signaling. RSVP signaling is soft state, which means the signaling process is frequently refreshed (by default once every 30 seconds) since the forwarding path across the IP network may change and therefore information about the QoS and capacity requested by a flow must be communicated periodically. This so-called soft-state mode of operation creates an additional processing load on a router even greater than that of an ATM switch. Furthermore, if the processor of a boundary router is overloaded by a large number of invalid RSVP requests, the processor may crash, thereby disrupting service for all flows for all customers being handled by the router with the failing processor.

[0009] In recognition of the problems associated with implementing admission control utilizing conventional Intserv RSVP signaling, the IETF promulgated the Differentiated Services (Diffserv or DS) protocol defined in RFC 2475. Diffserv is an IP QoS architecture that achieves scalability by conveying an aggregate traffic classification within a DS field (e.g., the IPv4 Type of Service (TOS) byte or IPv6 traffic class byte) of each IP-layer packet header. The first six bits of the DS field encode a Diffserv Code Point (DSCP) that requests a specific class of service or Per Hop Behavior (PHB) for the packet at each node along its path within a Diffserv domain.

[0010] In a Diffserv domain, network resources are allocated to aggregates of packet flows in accordance with service provisioning policies, which govern DSCP marking and traffic conditioning upon entry to the Diffserv domain and traffic forwarding within the Diffserv domain. The marking (i.e., classification) and conditioning operations need be implemented only at Diffserv network boundaries. Thus, rather than requiring end-to-end signaling between the transmitter and receiver to establish a flow having a specified QoS, Diffserv enables an ingress boundary router to

provide the QoS to aggregated flows simply by examining and/or marking each IP packet's header.

[0011] Although the Diffserv standard addresses Intserv scalability limitation by replacing Intserv's processing-intensive signaling with a simple per packet marking operation that can easily be performed in hardware, implementation of the Diffserv protocol presents a different type of problem. In particular, because Diffserv allows host marking of the service class, a Diffserv network customer link can experience a Denial of Service (DoS) attack if a number of hosts send packets to that link with the DS field set to a high priority. It should be noted that a set of hosts can exceed the subscribed capacity of a Diffserv service class directly by setting the DSCP or indirectly by submitting traffic that is classified by some other router or device to a particular DSCP. In Diffserv, an IP network can only protect its resources by policing at the ingress routers to ensure that each customer interface does not exceed the subscribed capacity for each Diffserv service class. However, this does not prevent a DoS attack.

[0012] **FIG. 2** depicts a DOS attack scenario in an exemplary IP network **10'** that implements the conventional Diffserv protocol. In **FIG. 2**, a number of ingress nodes (e.g., ingress boundary routers) **12***b*'-**12***d*' each admit traffic targeting a single link of an egress node (e.g., egress boundary router) **12***a*'. Although each ingress nodes **12'** polices incoming packets to ensure that customers do not exceed their subscribed resources at each DSCP, the aggregate of the admitted flows exceeds the capacity X of egress Link **1** of node **12***a*', resulting in a denial of service to the customer site served by this link.

SUMMARY OF THE INVENTION

[0013] In view of the limitations attendant to conventional implementations of the Intserv and Diffserv standards, the present invention recognizes that it would be useful and desirable to provide a method, system and apparatus for data communication that support a communication protocol that, unlike conventional Intserv implementations, is highly scalable and yet protects against the DoS attacks to which conventional Diffserv and other networks are susceptible.

[0014] A network architecture in accordance with the present invention includes a communication network that supports one or more network-based Virtual Private Networks (VPNs). The communication network includes a plurality of boundary routers that are connected by access links to CPE edge routers belonging to the one or more VPNs. To prevent traffic from outside a customer's VPN (e.g., traffic from other VPNs or the Internet at large) from degrading the QoS provided to traffic from within the customer's VPN, the present invention gives precedence to intra-VPN traffic over extra-VPN traffic on each customer's access link through access link prioritization or access link capacity allocation, such that extra-VPN traffic cannot interfere with inter-VPN traffic. Granting precedence to intra-VPN traffic over extra-VPN traffic in this manner entails special configuration of network elements and protocols, including partitioning between intra-VPN and extra-VPN traffic on the physical access link and access network using layer **2** switching and multiplexing, as well as the configuration of routing protocols to achieve logical traffic separation between intra-VPN traffic and extra-VPN traffic at the VPN boundary routers and CPE edge routers. By configuring the access networks, the VPN boundary routers and CPE edge routers, and the routing protocols of the edge and boundary routers in this manner, the high-level service of DoS attack prevention is achieved.

[0015] Additional objects, features, and advantages of the present invention will become apparent from the following detailed written description.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself however, as well as a preferred mode of use, further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

[0017] **FIG. 1** depicts a conventional Integrated Services (Intserv) network that implements per-flow QoS utilizing RSVP;

[0018] **FIG. 2** illustrates a conventional Differentiated Services (Diffserv) network that implements QoS on aggregated traffic flows utilizing DSCP markings in each packet header and is therefore vulnerable to a Denial of Service (DoS) attack;

[0019] **FIG. 3** depicts an exemplary communication network that, in accordance with a preferred embodiment of the present invention, resists DoS attacks by partitioning allocation and/or prioritization of access capacity by reference to membership in Virtual Private Networks (VPNs);

[0020] **FIG. 4** illustrates an exemplary network architecture that provides a CPE-based VPN solution to the DoS attack problem;

[0021] **FIG. 5** is a more detailed block diagram of a QoS-aware CPE edge router that may be utilized within the network architectures depicted in **FIGS. 4 and 7**;

[0022] **FIG. 6A** is a more detailed block diagram of a QoS-aware boundary router without VPN function that may be utilized within the network architectures illustrated in **FIGS. 4 and 7**;

[0023] **FIG. 6B** is a more detailed block diagram of a QoS-aware boundary router having VPN function that may be utilized within the network architecture illustrated in **FIG. 4**;

[0024] **FIG. 7** illustrates an exemplary network architecture that provides a network-based VPN solution to the DoS attack problem; and

[0025] **FIG. 8** is a more detailed block diagram of a QoS-aware VPN boundary router that may be utilized within the network architecture depicted in **FIG. 7**.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

[0026] With reference again to the figures and, in particular, with reference to **FIG. 3**, there is depicted a high level block diagram of an exemplary network architecture **20** that, in accordance with the present invention, provides a scalable method of providing QoS to selected traffic while protecting

a Virtual Private Network (VPN) customer's access and trunk network links against DoS attacks. Similar to the prior art network illustrated in **FIG. 2**, network architecture **20** of **FIG. 3** includes a Diffserv network **21** having N service provider boundary routers (BRs) **22** that each have L access links. What is different in network architecture **20** is that Diffserv network **21** supports a plurality of VPN instances, of which two are shown in the figure as identified by the access links of boundary routers **22** coupled to CPE edge routers (ERs) for a first network service customer **24** and an ER for a second network service customer **25** at each of four sites, respectively identified by letters a through d. Each CPE ER provides network service to a customer's local area networks (LANs). The service provider network-based VPN may support many more customers than the two shown in this figure.

[0027] In the exemplary communication scenario depicted in **FIG. 3**, hosts within the LANs of the first VPN customer coupled to CPE edge routers **24***b*-**24***d*, those within a second VPN customer's LANs coupled to CPE edge routers **25***a*-**25***d*, as well as sites coupled to other unillustrated CPE edge routers linked to boundary routers **22***a*-**22***d*, may all transmit packet flows targeting the LAN coupled to the first VPN customer CPE edge router **24***a*. If the conventional Diffserv network of the prior art described above with respect to **FIG. 2** were implemented, the outgoing access link **1** of boundary router **22***a* coupled to CPE edge router **24***a* could be easily overwhelmed by the convergence of these flows, resulting in a DoS. However, in accordance with the present invention, Diffserv network **21** of **FIG. 3** prevents a DoS attack from sites outside the VPN by directing intra-VPN traffic to a first logical port **27** on physical access link **1** of boundary router **22***a*, while directing traffic from other VPNs or other sites to a second logical port **28** on physical access link **1** of boundary router **22***a*.

[0028] To prevent traffic from outside a customer's community of interest (e.g., traffic from other VPNs or the Internet at large) from degrading the QoS provided to traffic from within the customer's community of interest (e.g., traffic from other hosts in the same business enterprise), the present invention either prioritizes intra-VPN traffic over extra-VPN traffic, or allocates access link capacity such that extra-VPN traffic cannot interfere with inter-VPN traffic. In other words, as described in detail below, each boundary router **22** gives precedence on each customer's access link to traffic originating within the customer's VPN, where a VPN is defined herein as a collection of nodes coupled by a shared network infrastructure in which network resources and/or communications are partitioned based upon membership of a collection of nodes. Granting precedence to intra-VPN traffic over extra-VPN traffic in this manner entails special configuration of network elements and protocols, including partitioning of the physical access between intra-VPN and extra-VPN traffic using layer **2** multiplexing and the configuration of routing protocols to achieve logical traffic separation. In summary, the configuration of the CPE edge router, the access network, the network-based VPN boundary router and the routing protocols involved in the edge and boundary routers cooperate to achieve the high-level service of DoS attack prevention, as detailed below. Conventional Diffserv and CPE edger router IPsec-based IP VPN implementations, by contrast, do not segregate traffic destined for sites within the same VPN (i.e., intra-VPN traffic) and traffic sent from other regions of the Internet (i.e., extra-VPN traffic).

[0029] Referring now to FIGS. **4-8**, at least two classes of implementations of the generalized network architecture **20** depicted in **FIG. 3** are possible. In particular, a network in accordance with the present invention can be realized as a CPE-based VPN implementation, as described below with reference to FIGS. **4-6**, or as a network-based VPN implementation, as described below with reference to FIGS. **7-8**.

[0030] Referring first to **FIG. 4**, there is illustrated an exemplary network architecture **30** that employs a CPE-based VPN to resist DoS attacks. The depicted network architecture includes a Diffserv-enabled IP VPN network **44**, a best effort IP public network **46**, and a plurality of customer Local Area Networks (LANs) **32**. Customer LANs **32** each include one or more hosts **48** that can function as a transmitter and/or receiver of packets communicated over one or both of networks **44** and **46**. In the exemplary implementation illustrated in **FIG. 4**, it is assumed that customer LANs **32***a* and **32***b* belong to the same community of interest (i.e., VPN), such as a business enterprise.

[0031] Each customer LAN **32** is coupled by a respective CPE edge router **34** and physical access link **35** to a respective access network (e.g., an L2 access network) **38**. Access networks **38***a* and **38***b* each have a first L2 access logical connection to a boundary router (BR) **40** of Diffserv-enabled IP VPN network **44** and a second L2 access logical connection to a boundary router (BR) **42** of best effort IP public network **46**. As illustrated in **FIG. 4** by differing line styles representing intra-VPN and extra-VPN traffic, VPN-aware CPE edge routers **34***a* and **34***b* route only packets with IP address prefixes belonging to the IP VPN via Diffserv-enabled IP VPN network **44**, and route all other traffic via best effort IP public network **46**. To enhance security of customer LANs **32**, CPE edge routers **34***a* and **34***b* send all traffic to and from best effort IP public network **46** through a respective one of firewalls **36***a* and **36***b*.

[0032] In the network architecture illustrated in **FIG. 4**, DoS attacks originating outside of the IP VPN are prevented by configuration of boundary routers **40***a*-**40***b* and **42***a*-**42***b* to appropriately utilize the two logical connections of access networks **38***a* and **38***b* to grant precedence to intra-VPN traffic. For example, in a first configuration, a higher priority is assigned to the L2 access logical connection with Diffserv-enabled IP VPN network **44** than to the L2 access logical connection with best effort public IP network **46**. L2 access networks that support such prioritization of access links **35** include Ethernet (e.g., utilizing Ethernet priority), ATM (e.g., utilizing ATM service categories), and many frame relay (FR) network implementations. These implementations can each be provisioned utilizing well-known techniques. With this configuration, each boundary router **40** of Diffserv enabled IP VPN network **44** shapes the transmission rate of packets to its logical connection to access network **38** to a value less than that of the access link to prevent starvation of the L2 access logical connection to best effort IP public network **46**. Alternatively, in a second configuration, boundary routers **40***a*-**40***b* and **42***a*-**42***b* may be individually configured to shape the traffic destined for each L2 access network logical connection to a specified rate, where the sum of these rates is less than or equal to the

transmission capacity of the physical access medium linking CPE edge routers **34** and access networks **38**. In either of these alternative configurations, boundary routers **40** and **42** perform scheduling and prioritization based upon packets' DSCP markings and shape to the capacity allocated to the access network connection for IP VPN traffic.

[0033] As will be appreciated by those skilled in the art, selection of which of the alternative configurations to implement is a matter of design choice, as each configuration has both advantages and disadvantages. For example, with the first configuration, coordination of the access network configuration between networks **44** and **46** is easier. However, if access networks **38** implement only strict priority, then IP VPN traffic from Diffserv-enabled IP VPN network **44** may starve best effort traffic communicated over IP public network **46**. The second configuration addresses this disadvantage by allocating a portion of the access link capacity to each type of network access (i.e., both intra-VPN and extra-VPN). However, if boundary routers **40** and **42** shape traffic in accordance with the second configuration, unused access capacity to one of networks **44** and **46** cannot be used to access the other network. That is, since the shapers are on separate boundary routers **40** and **42**, only non-work-conserving scheduling is possible.

[0034] With reference now to **FIG. 5**, there is illustrated a more detailed block diagram of a QoS-aware CPE edge router **34** that may be utilized within the network architecture depicted in **FIG. 4**. As illustrated, CPE edge router **34** includes a number of LAN ports **60**, which provide connections for a corresponding number of customer LANs **32**. For example, in **FIG. 5**, LAN port **60***a* is connected to a customer LAN **32** including a number of hosts **48** respectively assigned 32-bit IP addresses "a.b.c.d,""a.b.c.e.," and "a.b.c.f."

[0035] Each LAN port is also coupled to a forwarding function **62**, which forwards packets between LAN ports **60** and one or more logical ports (LPs) **66** residing on one or more Wide Area Network (WAN) physical ports **64** (only one of which is illustrated). LPs **66**, which each comprise a layer-2 sub-interface, may be implemented, for example, as an Ethernet Virtual LAN (VLAN), FR Data Link Connection Identifier (DLCI), ATM Virtual Channel Connection (VCC), or Point-to-Point Protocol (PPP)/High-Level Data Link Control (HDLC) running on a Time Division Multiplexed (TDM) channel. WAN physical port **64** employs a scheduler **68** to multiplex packets from logical ports **64** onto the transmission medium of an access network **38** and forwards packets received from access network **38** to the respective logical port utilizing a forwarding function **70**.

[0036] When a LAN port **60** of CPE edge router **34** receives packets from a customer LAN **32**, the packets first pass through a classifier **80**, which determines by reference to a classifier table **82** how each packet will be handled by CPE edge router **34**. As illustrated in **FIG. 5**, classifier table **82** may have a number of indices, including Source Address (SA) and Destination Address (DA), Source Port (SP) and Destination Port (DP), Protocol Type (PT), DSCP, or other fields from packets' link, network or transport layer headers. Based upon a packet's values for one or more of these indices, classifier **72** obtains values for a policer (P), marker (M), destination LP, and destination LP queue (Q) within CPE edge router **34** that will be utilized to process the packet. In alternative embodiments of the present invention, lookup of the destination LP and destination LP queue entries could be performed by forwarding function **62** rather than classifier **80**.

[0037] As shown, table entry values within classifier table **82** may be fully specified, partially specified utilizing a prefix or range, or null (indicated by "-"). For example, the SAs of hosts **48** of LAN **32** are fully specified utilizing 32-bit IP addresses, DAs of several destination hosts are specified utilizing 24-bit IP address prefixes that identify particular IP networks, and a number of index values and one policing value are null. In general, the same policer, marker, and/or shaper values, which for Intserv flows are taken from RSVP RESV messages, may be specified for different classified packet flows. For example, classifier table **82** specifies that policer P1 and marker M1 will process packets from any SA marked with DSCP "101" as well as packets having a SA "a.b.c.e" marked with DSCP "010." However, classifier table **82** distinguishes between flows having different classifications by specifying different destination LP values for traffic having a DA within the VPN (i.e., intra-VPN traffic) and traffic addressed to hosts elsewhere in the Internet (i.e., extra-VPN traffic). Thus, because IP address prefixes "r.s.t,""w.x.y," and "l.m.n" all belong to the same VPN as network **32**, traffic matching these DAs is sent via LP-1 **66***a* to other sites within the same VPN over the Diffserv-enabled IP VPN network **44** while all other traffic is sent via LP-2 **66***b* to best effort IP public network **46**.

[0038] The logical port **66** and LP queue to which packets are forwarded can be determined by static configuration or dynamically by a routing protocol. In either case, a VPN route should always have precedence over an Internet route if a CPE router **34** has both routes installed for the same destination IP address. Such priority can be achieved in any of several ways, including (1) use of Interior Gateway Protocol (IGP) (i.e., OSPF and IS-IS) to install VPN routes and EBGP or static routing to install Internet routes or (2) use of EBGP to install both VPN routes and Internet routes, with a higher local preference being given for VPN routes.

[0039] After classification, packets are policed and marked, as appropriate, by policers P0, P1 and markers M0, M1, M2 as indicated by classifier table **82** and then switched by forwarding function **62** to either logical port **66***a* or **66***b*, as specified by the table lookup. Within the specified logical port **66**, packets are directed to the LP queues Q0-Q02 specified by classifier table **82**. LP queues Q0-Q2 perform admission control based upon either available buffer capacity or thresholds, such as Random Early Detection (RED). A scheduler **90** then services LP queues Q0-Q2 according to a selected scheduling algorithm, such as First In, First Out (FIFO), Priority, Weighted Round Robin (WRR), Weighted Fair Queuing (WFQ) or Class-Based Queuing (CBQ). For example, in the illustrated embodiment, scheduler **90** of LP-2 **66***a* implements WFQ based upon the weight w_i associated with each LP queue i and the overall WFQ scheduler rate r_2 for logical port **2**, thereby shaping traffic to the rate r_2. Finally, as noted above, scheduler **68** of physical WAN port **64** services the various logical ports **66** to control the transmission rate to access network **38**.

[0040] CPE edge router **34** receives packets from access network **38** at WAN physical port **64** and then, utilizing

forwarding function **70**, forwards packets to the appropriate logical port **66***a* or **66***b* as indicated by configuration of access network **38** as it maps to the logical ports. At each logical port **66**, packets pass through a classifier **100**, which generally employs one or more indices within the same set of indices discussed above to access a classifier table **102**. In a typical implementation, the lookup results of classifiers **100** are less complex than those of classifier **80** because policing and marking are infrequently required. Thus, in the depicted embodiment, packets are forwarded by forwarding function **62** directly from classifiers **100** of logical ports **66** to the particular queues Q0-Q2 of LAN port **60***a* specified in the table lookup based upon the packets' DSCPs. As described above, queues Q0-Q2 of LAN port **60***a* are serviced by a scheduler **102** that implements WFQ and transmits packets to customer LAN **32**.

[0041] Referring now to **FIG. 6A**, there is depicted a more detailed block diagram of a QoS-aware boundary router without any VPN function, which may be utilized within the network architecture of **FIG. 4**, for example, to implement boundary routers **42**. As shown, boundary router **42** of **FIG. 6A** includes a plurality of physical ports **116**, a plurality of logical ports **110** coupled to access network **38** by a forwarding function **112** for incoming packets and a scheduler **114** for outgoing packets, and a forwarding function **118** that forwards packets between logical ports **110** and physical ports **116**. The implementation of multiple physical ports **116** permits fault tolerant connection to network core routers, and the implementation of multiple logical ports coupled to access network **38** permits configuration of one logical port (i.e., LP-1 **110***a*) as a Diffserv-enabled logical port and a second logical port (i.e., LP-2 **110***b*) as a best-effort logical port.

[0042] Thus, for traffic communicated from access network **38** through LP-2 **110***b* of boundary router **42** towards the network core, classifier **124** of LP-2 **110***b* directs all packets to marker M0 in accordance with classifier table **126**. Marker M0 remarks all packets received at LP-2 **110***b* with DSCP 000, thus identifying the packets as best-effort traffic. Classifier **120** of LP-1 **110***a*, by contrast, utilizes classifier table **122** to map incoming packets, which have already received DSCP marking at a trusted CPE (e.g., service provider-managed CPE edge router **34**), into queues Q0-Q2 on PHY-1 **116***a*, which queues are each associated with a different level of QoS. Because the packets have already been multi-field classified, marked and shaped by the trusted CPE, boundary router **42** need not remark the packets. If, however, the sending CPE edge router were not a trusted CPE, boundary router **42** would also need to remark and police packets received at LP-1 **110***a*.

[0043] Following classification (and marking in the case of traffic received at LP-2 **110***b*), traffic is forwarded to an appropriate physical port **116** or logical port **110** by forwarding function **118**. In contrast to edge router **34** of **FIG. 5**, which utilizes classifiers to perform the full forwarding lookup, boundary router **42** employs an alternative design in which forwarding function **118** accesses forwarding table **128** with a packet's DA to determine the output port, It namely, LP-1 **110***a*, LP-2 **110***b*, or PHY-1 **116***a* in this example. In the case of a non-VPN router, forwarding table **128** is populated by generic IP routing protocols (e.g., Border Gateway Protocol (BGP)) or static configuration (e.g., association of the 24-bit IP address prefix "d.e.f." with LP-2 **110***b*). An alternative implementation could centrally place the IP lookup forwarding function in forwarding function **62**. The exemplary implementation shown in **FIG. 6** assumes that boundary router **42** sends all traffic bound for the network core to only one of the physical ports **116** connected to a core router. In other embodiments, it is possible, of course, to load balance traffic across physical ports **116**. In addition, implementations omitting the core router or employing one or more logical ports to one or more core routers are straightforward extensions of the depicted design.

[0044] For traffic communicated to access network **38** through boundary router **42**, classifier **132** accesses classifier table **134** utilizing the DSCP of the packets to direct each packet to the appropriate one of queues Q0-Q-2 for the QoS indicated by the packet's DSCP. For a customer that has purchased a Diffserv-enabled logical port **110**, this has the effect of delivering the desired QoS since the source CPE has policed and marked the flow with appropriate DSCP value. Although a best-effort customer is capable of receiving higher quality traffic, preventing such a one-way differentiated service would require significant additional complexity in the classifier and include distribution of QoS information via routing protocols to every edge router in a service provider network.

[0045] With reference now to **FIG. 6B**, there is depicted a more detailed block diagram of a QoS-aware VPN boundary router **40**, which may be utilized to provide Diffserv-enabled and DoS-protected VPN service within the network architecture depicted in **FIG. 4**. As shown, boundary router **40** includes a plurality of physical ports **226** for connection to core routers of Diffserv-enabled IP VPN network **44**, a plurality of Diffserv-enabled logical ports **224** coupled to an access network **38** by a forwarding function **220** for incoming packets and a scheduler **222** for outgoing packets, and a forwarding function **228** that forwards packets between logical ports **224** and physical ports **226**.

[0046] Each Diffserv-enabled logical port **224** implemented on boundary router **40** serves a respective one of a plurality of VPNs. For example, Diffserv-enabled logical port LP-A **224***a* serves a customer site belonging to VPN A, which includes customer sites having the 24-bit IP address prefixes "a.b.c." and "a.b.d." Similarly, Diffserv-enabled logical port LP-B **224***b* serves a customer site belonging to VPN B, which includes two customer sites having the 24-bit IP address prefixes "b.c.d." and "b.c.c." Diffserv-enabled logical ports **224** do not serve sites belonging to best effort IP public network **46** since such traffic is routed to boundary routers **42**, as shown in **FIG. 4**.

[0047] As further illustrated in **FIG. 6B**, each core-facing physical port **226** of boundary router **40** is logically partitioned into a plurality of sub-interfaces implemented as logical tunnels **240**. As will be appreciated by those skilled in the art, a tunnel may be implemented utilizing any of a variety of techniques, including an IP-over-IP tunnel, a Generic Routing Encapsulation (GRE) tunnel, an IPsec operated in tunnel mode, a set of stacked Multi-Protocol Label Switching (MPLS) labels, a Layer 2 Tunneling Protocol (L2TP), or a null tunnel. Such tunnels can be distinguished from logical ports in that routing information for multiple VPNs can be associated with a tunnel in a nested manner. For example, in the Border Gateway Protocol

(BGP)/MPLS VPNs described in IETF RFC 2547, the topmost MPLS label determines the destination boundary router while the bottommost label determines the destination VPN.

[0048] In operation, a classifier **230** on each of Diffserv-enabled logical ports **224** classifies packets flowing from access network **38** through boundary router **40** to the network core of Diffserv-enabled IP VPN network **44** in accordance with the packets' DSCP values by reference to a respective classifier table **232**. As depicted, classifier tables **232***a* and **232***b* are accessed utilizing the DSCP as an index to determine the appropriate one of queues Q0-Q2 on physical port PHY-1 **226***a* for each packet. Packets received by physical ports **226** are similarly classified by a classifier **250** by reference to a classifier table **254** to determine an appropriate one of queues Q0-Q2 for each packet on one of logical ports **224**. After classification (and optional (re)marking as shown at LP-B **224***b*), forwarding function **228** switches packets between logical ports **224** and physical ports **226** by reference to VPN forwarding tables **234***a*-**234***n*, which are each associated with a respective VPN. Thus, for example, VPN forwarding table **234***a* provides forwarding routes for VPN A, and VPN forwarding table **234***b* provides forwarding routes for VPN B.

[0049] VPN forwarding tables **234** are accessed utilizing the source port and DA as indices. For example, in the exemplary network configuration represented in forwarding table **234***a*, traffic within VPN A addressed with a DA having a 24-bit IP address prefix of "a.b.d." traverses TNL-1 **240***a*, and traffic received at TNL-1 **240***b* is directed to LP-A **224***a*. Similar routing between TNL-2 **240***b* and LP-B **224***b* can be seen in VPN routing table **234***b*. As discussed above, VPN forwarding tables **234** can be populated by static configuration or dynamically utilizing a routing protocol.

[0050] Following processing by forwarding function **178**, packets are each directed to the output port queue corresponding to their DSCP values. For example, packets marked with the QoS class associated with DSCP 101 are placed in Q2, packets marked with the QoS class associated with DSCP 010 are placed in Q1, and traffic marked with DSCP 000 is placed in Q0. Schedulers **236** and **252** then schedule output of packets from queues Q0-Q2 to achieve the requested QoS.

[0051] With reference now to **FIG. 7**, there is illustrated an exemplary network architecture **150** that provides a network-based VPN solution to the DoS attack problem. In **FIG. 7**, like reference numerals and traffic notations are utilized to identify features corresponding to features of network architecture **30** depicted in **FIG. 4**.

[0052] As depicted, network architecture **150** of **FIG. 7**, like network architecture **30** of **FIG. 4**, includes a Diffserv-enabled IP VPN network **44**, a best effort IP public network **46**, and a plurality of customer Local Area Networks (LANs) **32**. As above, customer LANs **32***a* and **32***b* belong to the same VPN and each include one or more hosts **48** that can function as a transmitter and/or receiver of packets. Each customer LAN **32** is coupled by a CPE edge router **34** and a physical access link **153** to a respective access network (e.g., an L2 or L3 access network) **154**. In contrast to access networks **38** of **FIG. 4**, which have separate logical connections for QoS and best effort traffic, access networks **154** are only connected to boundary routers **156** of Diffserv-enabled IP VPN network **44**, which have separate logical connections to boundary routers **42** of best effort IP public network **46**. Thus, intra-VPN traffic destined for network **44** and extra-VPN traffic destined for network **46** are both routed through boundary routers **156**, meaning that work-conserving scheduling between the two classes of traffic is advantageously permitted. However, as a consequence, the complexity of boundary routers **156** necessarily increases because each boundary router **156** must implement a separate forwarding table for each attached customer, as well as a full Internet forwarding table that can be shared among customers.

[0053] Referring now to **FIG. 8**, there is depicted more detailed block diagram of a QoS-aware VPN boundary router in which the policers, shapers, schedulers, logical port access network connections and forwarding tables are configured to provide Diffserv-enabled and DoS-protected VPN service within the network architecture depicted in **FIG. 7**. As shown, boundary router **156** includes a plurality of physical ports **176** for connection to network core routers, a plurality of Diffserv-enabled logical ports **174** coupled to access network **154** by a forwarding function **170** for incoming packets and a scheduler **172** for outgoing packets, and a forwarding function **178** that forwards packets between logical ports **174** and physical ports **176**.

[0054] Because each CPE edge router **34** is coupled to a boundary router **156** by only a single access link through access network **154**, each network customer site is served at boundary router **156** by a pair of Diffserv-enabled logical ports **174**, one for intra-VPN traffic and one for extra-VPN traffic. For example, Diffserv-enabled logical ports LP-A1 **174***a* and LP-A2 **174** serve a single customer site belonging to VPN A, which includes at least two customer sites having the 24-bit IP address prefixes "a.b.c." and "a.b.d." In the depicted embodiment, LP-A1 **174***a* provides access to QoS traffic communicated across Diffserv-enabled IP VPN network **44** to and from sites belonging to VPN A, while LP-A2 **174***b* provides access to best effort traffic to and from best effort IP public network **46**.

[0055] As further illustrated in **FIG. 8**, each core-facing physical port **176** of boundary router **156** is logically partitioned into a plurality of sub-interfaces implemented as logical tunnels **180**. As will be appreciated by those skilled in the art, a tunnel may be implemented utilizing any of a variety of techniques, including an IP-over-IP tunnel, a Generic Routing Encapsulation (GRE) tunnel, an IPsec operated in tunnel mode, a set of stacked Multi-Protocol Label Switching (MPLS) labels, or a null tunnel. Such tunnels can be distinguished from logical ports in that routing information for multiple VPNs can be associated with a tunnel in a nested manner. For example, in the Border Gateway Protocol (BGP)/MPLS VPNs described in IETF RFC 2547, the topmost MPLS label determines the destination boundary router while the bottommost label determines the destination VPN.

[0056] In operation, a classifier **182** on each of Diffserv-enabled logical ports **174** classifies packets flowing from access network **154** through boundary router **156** to the network core in accordance with the packets' DSCP values by reference to a respective classifier table **190**. As depicted, classifier tables **190***a* and **190***b* are accessed utilizing the DSCP as an index to determine the appropriate one of

queues Q0-Q2 on physical port PHY-1 **176***a* for each packet. Packets received by physical ports **176** are similarly classified by a classifier **198** by reference to a classifier table **192** to determine an appropriate one of queues Q0-Q2 for each packet on one of logical ports **174**. After classification (and optional (re)marking as shown at LP-A2 **174***b*), forwarding function **178** switches packets between logical ports **174** and physical ports **176** by reference to VPN forwarding tables **194***a*-**194***n*, which are each associated with a respective VPN and shared Internet forwarding table **195**. Thus, for example, forwarding table **194***a* contains entries providing forwarding routes for VPN A, while Internet forwarding table **195** contains entries providing forwarding routes for packets specifying LP-A2 or TNL-2 (i.e., the logical interfaces configured for Internet access) as a source.

[0057] Forwarding tables **194** are accessed utilizing the source port and DA as indices. For example, in the exemplary network configuration represented in forwarding table **194***a*, intra-VPN traffic addressed with a DA having a 24-bit IP address prefix of "a.b.d." traverses TNL-1 **180***a*, while extra-VPN (i.e., Internet) traffic traverses TNL-2 **180***b* (which could be a null tunnel). Forwarding table **194***a* further indicates that intra-VPN traffic received via TNL-1 **180***a* is directed to LP-A1 **174***a*, and all other traffic arriving from the Internet via tunnel TNL-2 **180***b* addressed with a DA having a 24-bit IP address prefix of "a.b.c." is sent to LP-A2 **174***b*. Traffic that terminates to other ports on boundary router **156** (i.e., traffic having a Local DA) is sent to other ports of boundary router **156** (indicated as LP-x). In other words, the entries in forwarding table **194***a* marked "Local" specify address prefixes other than those assigned to VPNs (e.g., a.b.c/24) that are assigned to interfaces on boundary router **156**.

[0058] Following processing by forwarding function **178**, packets are each directed to the output port queue corresponding to their DSCP values. For example, packets marked with the QoS class associated with DSCP 101 are placed in Q2, packets marked with the QoS class associated with DSCP 010 are placed in Q1, and best effort traffic marked with DSCP 000 is placed in Q0. Schedulers **196** then schedule output of packets from queues Q0-Q2 to achieve the requested QoS.

[0059] As has been described, the present invention provides an improved network architecture for providing QoS to intra-VPN traffic while protecting such flows against DoS attack from sources outside the VPN. The present invention provides DoS-protected QoS to selected flows utilizing a network-based VPN service and a best effort Internet service connected to a CPE edge router using a L2 access network with appropriately configured routing protocols. Diffserv marking at the edge and handling in the network-based VPN core provides QoS to selected flows while logically partitioning intra-VPN and extra-VPN traffic to prevent DoS to a VPN network customer site due to traffic originating from outside of the customer's VPN exceeding that site's access capacity. Even further protection from traffic originating from within the customer's VPN is possible using Intserv policy control, implemented on the CPE edge router and/or the QoS-aware boundary router, as described in IETF RFC 2998, incorporated herein by reference.

[0060] The network architecture of the present invention may be realized in CPE-based and network-based implementations. The CPE-based implementation permits easy configuration of the access networks linking the CPE edge routers and service provider boundary routers and permits QoS to be offered to VPN sites without implementing Diffserv across the entire service provider network. The network-based configuration advantageously permits work conserving scheduling that permits extra-VPN traffic to utilize excess access capacity allocated to intra-VPN traffic.

[0061] While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents. For example, although the present invention has been described with respect to preferred embodiments in which network-based VPNs are implemented within a Diffserv network, it should be understood that the present invention is not restricted to use with Diffserv networks, but is instead to other network-based VPNs, which may be implemented, for example, utilizing BGP/MPLS as taught in RFC 2547 or virtual routers as taught in RFC 2917. In addition, although **FIGS. 3, 4** and **7** illustrate the connection of each CPE edge router to a VPN network and a best effort network by one access link, it should be understood that, for redundancy, a CPE edge router may be connected by multiple access links to one or more access networks, which provide logical connections to one or more boundary routers of each of the VPN and best effort networks. In such "dual homing" implementations, the multiple access links can be utilized in either a primary/backup or load-sharing arrangement through installation of static routes in the service provider boundary routers or dynamic configuration of the service provider boundary routers utilizing routing protocols (e.g., EBGP). This would require that the CPE edge router implement multiple forwarding tables and separate instances of the routing protocol for the VPN and Internet access address spaces. The implementation of such a CPE edge router would be similar to that illustrated in **FIG. 8** and described in the associated text, with only a single VPN table and a single table for Internet routes.

What is claimed is:

1. A virtual private network (VPN)-aware CPE edge router, comprising:

at least one customer network port having a connection for a customer network belonging to a VPN;

at least one physical port on which at least first and second logical ports reside, said physical port having a physical port scheduler that schedules transmission of packets from said first and second logical ports onto a physical access link, wherein said physical port scheduler ensures access to said physical access link by outgoing traffic from said first logical port by one of (1) access link capacity allocation between outgoing traffic from said first and second logical ports and (2) access link prioritization of outgoing traffic from said first logical port over outgoing traffic from said second logical port; and

a forwarding function that forwards to said first logical port only packets identified as intra-VPN traffic to be

communicated to a destination host belonging to the VPN and forwards other packets to said second logical port.

2. The virtual private network (VPN)-aware CPE edge router of claim 1, wherein:

said customer network port further includes a plurality of markers that each mark packets with a respective one of a plurality of service markings that each specify a different one of a plurality of qualities of service;

at least said first logical port includes a plurality of queues each associated with a respective one of said plurality of qualities of service and a logical port scheduler that schedules transmission of packets from said plurality of queues; and

said forwarding function places packets in particular ones of said plurality of queues in accordance with marking of said packets by said plurality of markers.

3. The virtual private network (VPN)-aware CPE edge router of claim 2, wherein each of said plurality of markers marks packets by setting a Differentiated Services Code Point (DSCP) in an Internet Protocol header of each marked packet.

4. The virtual private network (VPN)-aware CPE edge router of claim 1, wherein:

said customer network port includes a plurality of queues each associated with a respective one of said plurality of qualities of service for outgoing packets destined for said customer network and a customer network port scheduler that schedules transmission of packets from said plurality of queues; and

said forwarding function places packets received at said plurality of logical ports in particular ones of said plurality of queues in accordance with marking of said packets.

5. The virtual private network (VPN)-aware edge router of claim 1, wherein said customer network port includes a classifier that classifies at least some packets as intra-VPN traffic based at least partially upon a source address and a partial destination address.

6. The virtual private network (VPN)-aware edge router of claim 5, wherein said classifier classifies at least some packets based at least partially upon a marking by a transmitter host.

7. The virtual private network (VPN)-aware edge router of claim 5, said at least one customer network port further including at least one policer, wherein said classifier sends packets to said at least one policer for policing in accordance with classification of said packets.

8. The virtual private network (VPN)-aware edge router of claim 5, said classifier having an associated classification table that associates particular values of packet header fields with particular ones of said first and second logical ports.

9. The virtual private network (VPN)-aware edge router of claim 1, wherein said physical port scheduler allocates a first portion of access link capacity to outgoing traffic from said first logical port and a second portion of access link capacity to outgoing traffic from said second logical port.

10. The virtual private network (VPN)-aware edge router of claim 1, wherein said physical port scheduler grants outgoing traffic from said first logical port a higher access link priority than outgoing traffic from said second logical port.

11. A network access system for use with an Internet protocol (IP) network infrastructure that implements a network-based Virtual Private Network (VPN) and a best effort network, said network access system comprising:

a VPN-aware customer premises equipment (CPE) edge router,

an access network supporting at least a first logical connection between the CPE edge router and network-based VPN and a second logical connection between the CPE edge router and the best effort network;

wherein said VPN-aware CPE edge router includes:

at least one customer network port having a connection for a customer network belonging to the VPN;

at least one physical port on which at least first and second logical ports reside, said physical port having a physical port scheduler that transmits outgoing packets from said first logical port via said first logical connection and transmits outgoing packets from said second logical port via said second logical connection, wherein said physical port scheduler ensures access to said physical access link by outgoing traffic from said first logical port by one of (1) access link capacity allocation between outgoing traffic from said first and second logical ports and (2) access link prioritization of outgoing traffic from said first logical port over outgoing traffic from said second logical port; and

a forwarding function configured to forward to said first logical port only packets identified as intra-VPN traffic to be communicated to a destination host belonging to the VPN and to forward other packets to said second logical port.

12. The network access system of claim 11, wherein:

said customer network port further includes a plurality of markers that each mark packets with a respective one of a plurality of service markings that each specify a different one of a plurality of qualities of service;

at least said first logical port includes a plurality of queues each associated with a respective one of said plurality of qualities of service and a logical port scheduler that schedules transmission of packets from said plurality of queues; and

said forwarding function places packets in particular ones of said plurality of queues in accordance with marking of said packets by said plurality of markers.

13. The network access system of claim 12, wherein each of said plurality of markers marks packets by setting a Differentiated Services Code Point (DSCP) in an Internet Protocol header of each marked packet.

14. The network access system of claim 11, wherein:

said customer network port includes a plurality of queues each associated with a respective one of said plurality of qualities of service for outgoing packets destined for said customer network and a customer network port scheduler that schedules transmission of packets from said plurality of queues; and

said forwarding function places packets received at said plurality of logical ports in particular ones of said plurality of queues in accordance with marking of said packets.

15. The network access system of claim 11, wherein said customer network port includes a classifier that classifies at least some packets as intra-VPN traffic based at least partially upon a source address and a partial destination address.

16. The network access system of claim 15, wherein said classifier classifies at least some packets based at least partially upon a marking by a transmitter host.

17. The network access system of claim 15, said at least one customer network port further including at least one policer, wherein said classifier sends packets to said at least one policer for policing in accordance with classification of said packets.

18. The network access system of claim 15, said classifier having an associated classification table that associates particular values of packet header fields with particular ones of said first and second logical ports.

19. The network access system of claim 11, wherein said physical port scheduler allocates a first portion of access link capacity to outgoing traffic from said first logical port and a second portion of access link capacity to outgoing traffic from said second logical port.

20. The network access system of claim 11, wherein said physical port scheduler grants outgoing traffic from said first logical port a higher access link priority than outgoing traffic from said second logical port.

21. The network access system of claim 11, wherein said access network comprises an L2 access network implemented with one of Asynchronous Transfer Mode, Ethernet and Frame Relay.

22. The network access system of claim 11, and further comprising the Internet protocol (IP) network infrastructure that implements a network-based Virtual Private Network (VPN) and a best effort network.

23. The network access system of claim 22, wherein said network-based VPN is implemented in a Differentiated Services domain.

24. A method of communicating data packets from a transmitter host toward a receiver host, said method comprising:

providing at least first and second logical ports on a physical port connected to an access link of an access network;

at a customer network port having a connection for a customer network belonging to a VPN, receiving one or more packet flows;

identifying packets in said one or more packet flows as intra-VPN traffic to be communicated to a destination host belonging to the VPN or as extra-VPN traffic;

forwarding only packets identified as intra-VPN traffic to the first logical port on the physical port and forwarding packets identified as extra-VPN traffic to said second logical port; and

protecting access to the access link by outgoing traffic from said first logical port by one of (1) access link capacity allocation between outgoing traffic from said first and second logical ports and (2) access link prioritization of outgoing traffic from said first logical port over outgoing traffic from said second logical port.

25. The method of claim 24, wherein:

said method further comprises marking packets at said customer network port with a respective one of a plurality of service markings that each specify a different one of a plurality of qualities of service;

said forwarding step includes placing packets in particular ones of a plurality of queues at said first logical port in accordance with marking of said packets; and

said method further comprises scheduling transmission of packets from said plurality of queues to achieve said plurality of qualities of service.

26. The method of claim 25, wherein said marking comprises setting a Differentiated Services Code Point (DSCP) in an Internet Protocol header of each marked packet.

27. The method of claim 24, wherein:

said forwarding step includes placing packets received at said plurality of logical ports in particular ones of a plurality of queues at said customer network port that each associated with a respective one of said plurality of qualities of service for outgoing packets destined for said customer network, said placing being performed in accordance with marking of said packets; and

said method further comprises scheduling transmission of packets from said plurality of queues to achieve said plurality of qualities of service.

28. The method of claim 24, and further comprising classifying, at said customer network port, at least some packets as intra-VPN traffic based at least partially upon a source address and a partial destination address.

29. The method of claim 28, and further comprising classifying at least some packets based at least partially upon a marking by a transmitter host.

30. The method of claim 28, policing packets at said customer network port in accordance with classification of said packets.

31. The method of claim 28, wherein said classifying comprises selecting one of said first and second logical ports as an output port for a packet by reference to a classification table that associates particular values of packet header fields with particular ones of said first and second logical ports.

32. The method of claim 24, wherein protecting access to the access link by outgoing traffic from said first logical port comprises allocating a first portion of access link capacity to outgoing traffic from said first logical port and a second portion of access link capacity to outgoing traffic from said second logical port.

33. The method of claim 24, wherein protecting access to the access link by outgoing traffic from said first logical port comprises granting outgoing traffic from said first logical port a higher access link priority than outgoing traffic from said second logical port.

* * * * *

US 20020067725A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0067725 A1

Oguchi et al. (43) Pub. Date: Jun. 6, 2002

(54) VIRTUAL NETWORK CONSTRUCTION METHOD, SYSTEM, AND RELAYING APPARATUS

(76) Inventors: **Naoki Oguchi**, Kawasaki (JP); **Yuji Saito**, Kawasaki (JP)

Correspondence Address:
ROSENMAN & COLIN LLP
575 MADISON AVENUE
NEW YORK, NY 10022-2585 (US)

(21) Appl. No.: 09/988,958

(22) Filed: Nov. 19, 2001

(30) Foreign Application Priority Data

Dec. 6, 2000 (JP) 2000-371913

Publication Classification

(51) Int. Cl.[7] H04L 12/28

(52) U.S. Cl. 370/390; 370/432

(57) ABSTRACT

In a virtual network construction method, a virtual network construction system, and a relaying apparatus within a public data communication network, control packets each having set a multicast address are multicast, and upon reception of the control packets by the relaying apparatuses belonging to the multicast address group, virtual links to the transmitting sources of the control packets are established and reply packets are returned through the virtual links, whereby the virtual links are established between all of the relaying apparatuses belonging to the multicast address group to establish the virtual network.

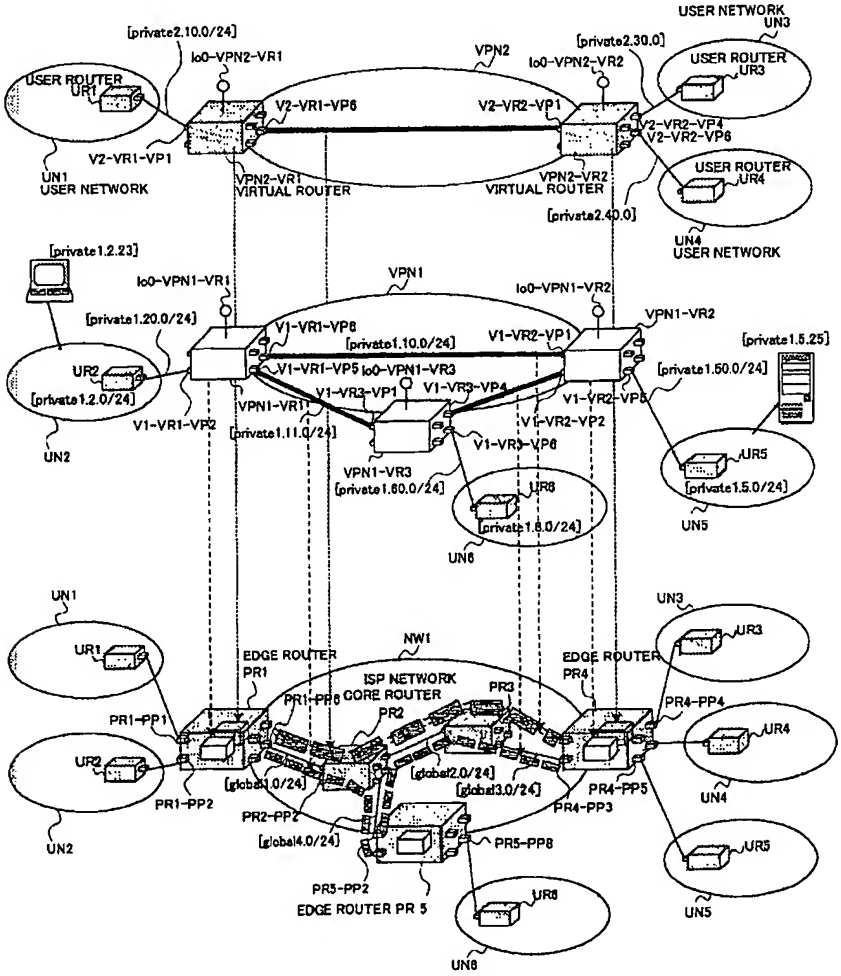


FIG.1

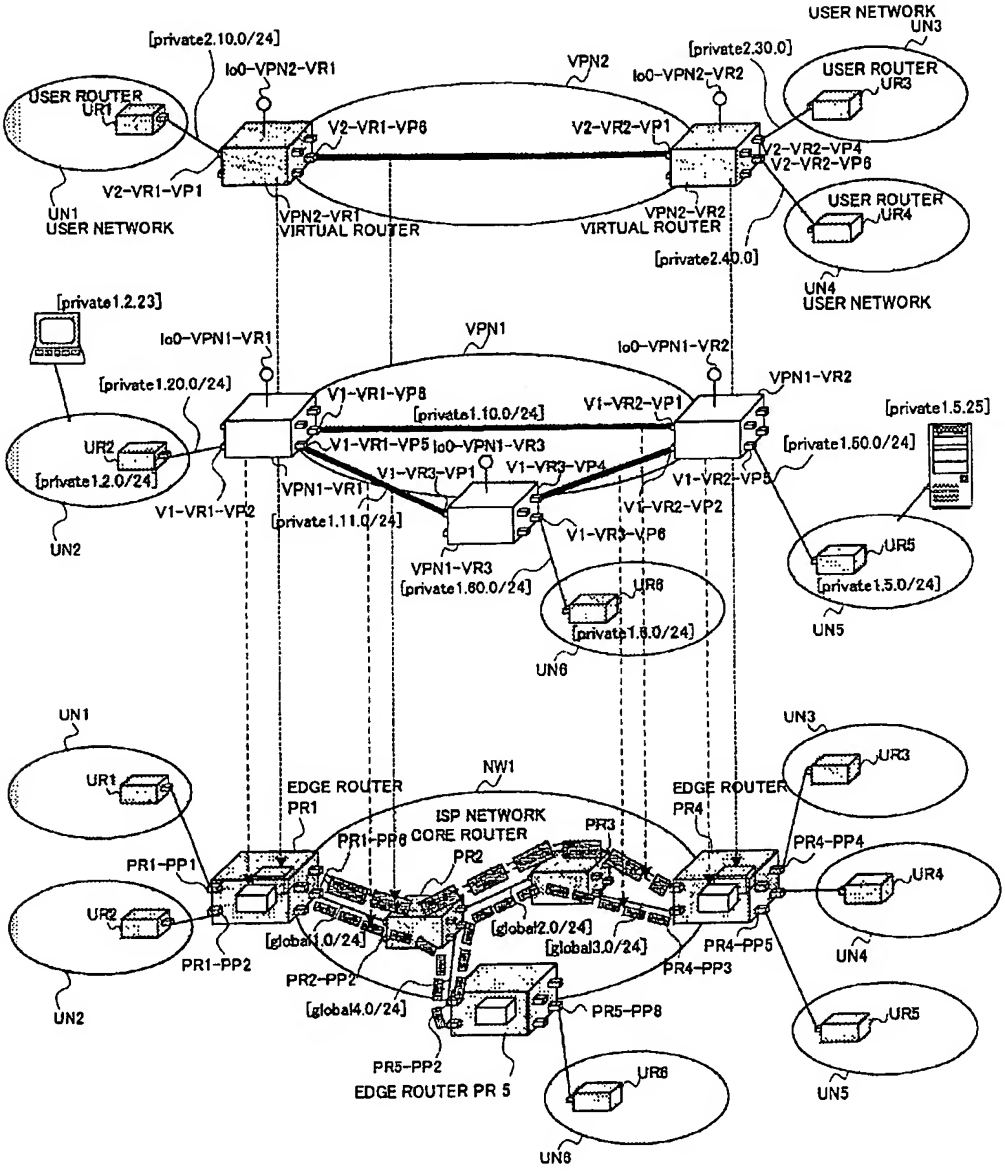

USER NETWORK
UN3
[private2.10.0/24]
lo0-VPN2-VR1
VPN2
[private2.30.0]
lo0-VPN2-VR2
USER ROUTER
UR1
USER ROUTER
UR3
V2-VR1-VP6
V2-VR2-VP1
V2-VR2-VP4
V2-VR2-VP6
V2-VR1-VP1
UN1
USER NETWORK
VPN2-VR1
VIRTUAL ROUTER
VPN2-VR2
VIRTUAL ROUTER
USER ROUTER
UR4
[private2.40.0]
UN4
USER NETWORK
[private1.2.23]
lo0-VPN1-VR1
VPN1
lo0-VPN1-VR2
VPN1-VR2
[private1.20.0/24]
V1-VR1-VP6
[private1.10.0/24]
V1-VR2-VP1
[private1.5.25]
UR2
V1-VR1-VP5
lo0-VPN1-VR3
[private1.60.0/24]
[private1.2.0/24]
V1-VR3-VP1
V1-VR3-VP4
VPN1-VR1
V1-VR2-VP5
V1-VR1-VP2
[private1.11.0/24]
V1-VR2-VP2
V1-VR3-VP6
UR5
UN2
VPN1-VR3
[private1.60.0/24]
UR6
[private1.5.0/24]
[private1.6.0/24]
UN5
UN6
UN1
UN3
UR1
EDGE ROUTER
PR1
NW1
ISP NETWORK
CORE ROUTER
EDGE ROUTER
PR4
UR3
PR1-PP6
PR3
PR4-PP4
PR1-PP1
PR2
UR4
UR2
[global1.0/24]
[global2.0/24]
[global3.0/24]
PR4-PP5
PR1-PP2
PR4-PP3
UN4
PR2-PP2
[global4.0/24]
PR5-PP8
UR5
PR5-PP2
EDGE ROUTER PR 5
UR6
UN2
UN5
UN6

FIG.2

VIRTUAL INTERFACE NAME	IP ADDRESS
V1-VR1-VP2	private1. 20. 1
V1-VR1-VP5	private1. 11. 1
V1-VR1-VP6	private1. 10. 1
V1-VR2-VP1	private1. 10. 2
V1-VR2-VP2	private1. 12. 2
V1-VR2-VP5	private1. 30. 1
V1-VR3-VP1	private1. 11. 2
V1-VR3-VP4	private1. 12. 1
V1-VR3-VP6	private1. 40. 1
lo0-VPN1-VR1	private1. 100. 1
lo0-VPN1-VR2	private1. 100. 2
lo0-VPN1-VR3	private1. 100. 3
lo0-VPN2-VR1	private2. 100. 1
lo0-VPN2-VR2	private2. 100. 2

FIG.3

INTERFACE NAME	IP ADDRESS
PR1-PP1	private2. 10. 1
PR1-PP2	private1. 20. 1
PR1-PP6	global1. 1
PR2-PP2	global1. 2
PR4-PP3	global3. 2
PR4-PP4	private2. 30. 1
PR4-PP5	private1. 30. 1
PR4-PP6	private2. 50. 1
PR5-PP2	global4. 2
PR5-PP8	private1. 40. 1
lo0-PR1	global100. 1
lo0-PR4	global100. 2
lo0-PR5	global100. 3

FIG.4

INTERFACE NAME	IP ADDRESS
UR1-PP1	private2. 10. 2
UR2-PP1	private1. 20. 2
UR3-PP1	private2. 30. 2
UR5-PP1	private2. 50. 2
UR6-PP1	private1. 30. 2

FIG.5

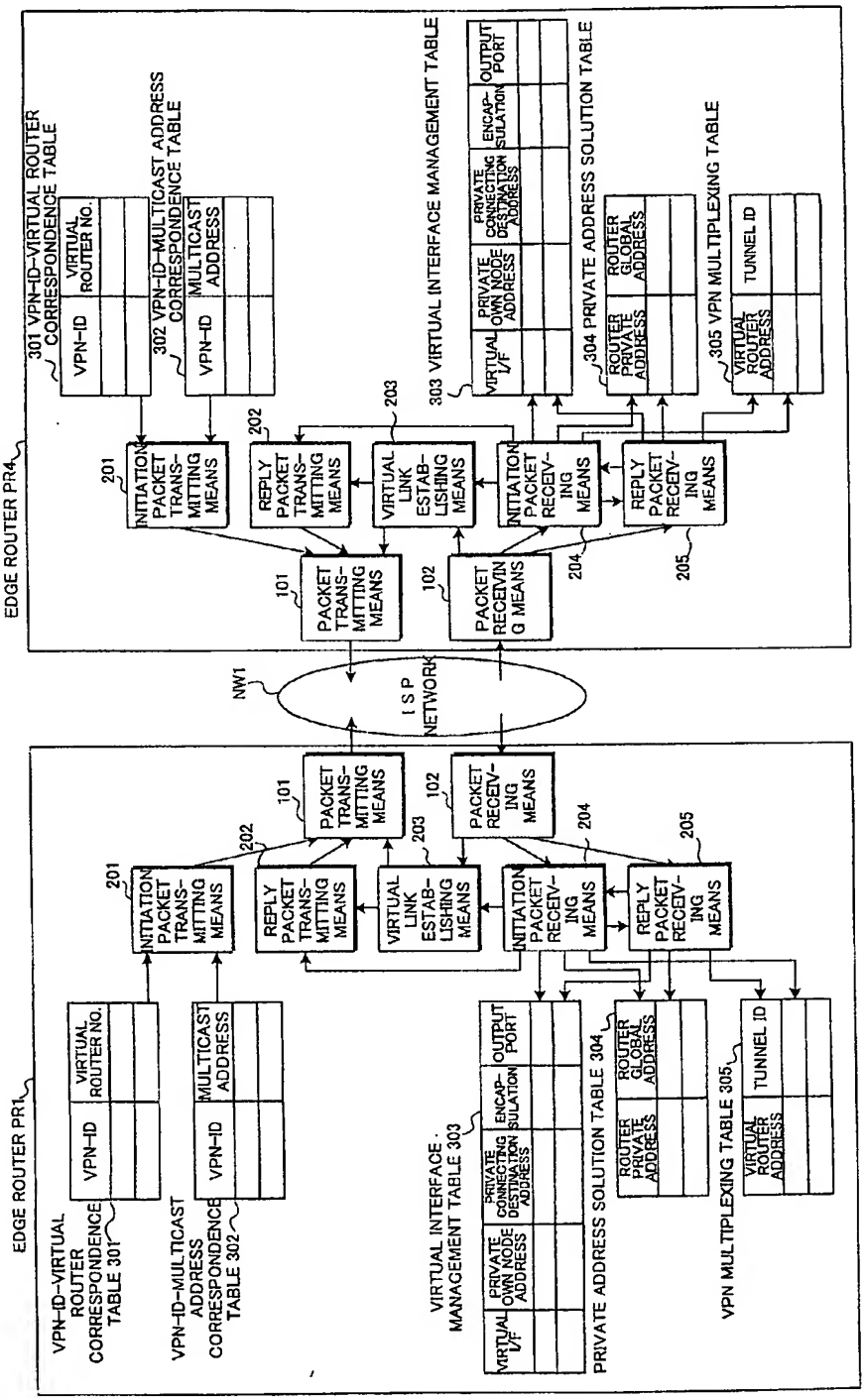

EDGE ROUTER PR4
EDGE ROUTER PR1
NW1
I S P NETWORK
PACKET TRANS-MITTING MEANS
PACKET RECEIVING MEANS
INITIATION PACKET TRANS-MITTING MEANS
REPLY PACKET TRANS-MITTING MEANS
VIRTUAL LINK ESTAB-LISHING MEANS
INITIATION PACKET RECEIV-ING MEANS
REPLY PACKET RECEIV-ING MEANS
301 VPN-ID-VIRTUAL ROUTER CORRESPONDENCE TABLE
VPN-ID
VIRTUAL ROUTER NO.
302 VPN-ID-MULTICAST ADDRESS CORRESPONDENCE TABLE
MULTICAST ADDRESS
303 VIRTUAL INTERFACE MANAGEMENT TABLE
VIRTUAL I/F
PRIVATE OWN NODE ADDRESS
PRIVATE CONNECTING DESTINATION ADDRESS
ENCAP-SULATION
OUTPUT PORT
304 PRIVATE ADDRESS SOLUTION TABLE
ROUTER PRIVATE ADDRESS
ROUTER GLOBAL ADDRESS
305 VPN MULTIPLEXING TABLE
VIRTUAL ROUTER ADDRESS
TUNNEL ID
VPN-ID-VIRTUAL ROUTER CORRESPONDENCE TABLE 301
VPN-ID-MULTICAST ADDRESS CORRESPONDENCE TABLE 302
VIRTUAL INTERFACE MANAGEMENT TABLE 303
PRIVATE ADDRESS SOLUTION TABLE 304
VPN MULTIPLEXING TABLE 305

FIG.6A EDGE ROUTER PR 1

VPN-ID	VIRTUAL ROUTER
1	VPN1-VR1
2	VPN2-VR1

FIG.6B EDGE ROUTER PR 4

VPN-ID	VIRTUAL ROUTER
1	VPN1-VR2
2	VPN2-VR2

FIG.7

VPN-ID	MULTICAST ADDRESS
1	239.192.0.1
2	239.192.0.2
...	...
1024	239.192.4.0

FIG.8

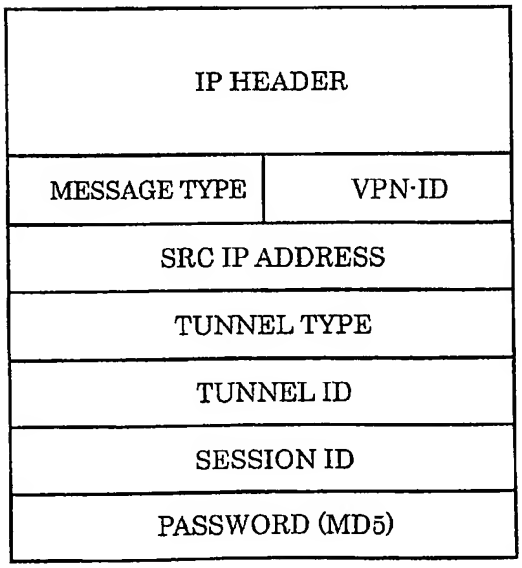


FIG.9

MESSAGE TYPE	0
VPN-ID	1
SRC IP ADDRESS	private1.100.1
TUNNEL TYPE	0(L2TP)
TUNNEL ID	0
SESSION ID	0
PASSWORD	11111

FIG.10A UPWARD TUNNEL (PR4→PR1)

TUNNEL ID	105
SESSION ID	200

FIG.10B DOWNWARD TUNNEL (PR1→PR4)

TUNNEL ID	300
SESSION ID	202

FIG.11

MESSAGE TYPE	1
VPN-ID	1
SRC IP ADDRESS	private1. 100. 2
TUNNEL TYPE	0 (L2TP)
TUNNEL ID	105
SESSION ID	200
PASSWORD	11111

FIG.12

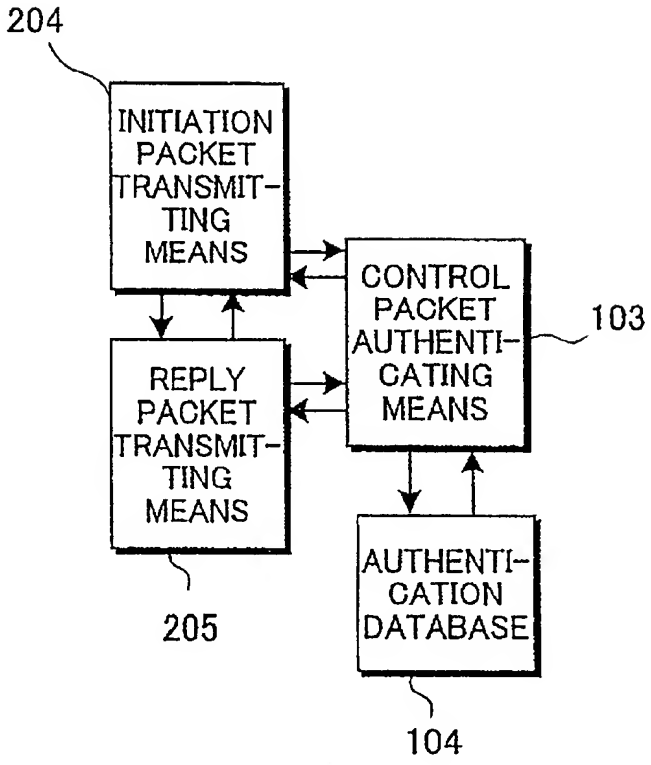

204
INITIATION PACKET TRANSMIT-TING MEANS
CONTROL PACKET AUTHENTI-CATING MEANS
103
REPLY PACKET TRANSMIT-TING MEANS
205
AUTHENTI-CATION DATABASE
104

FIG.13

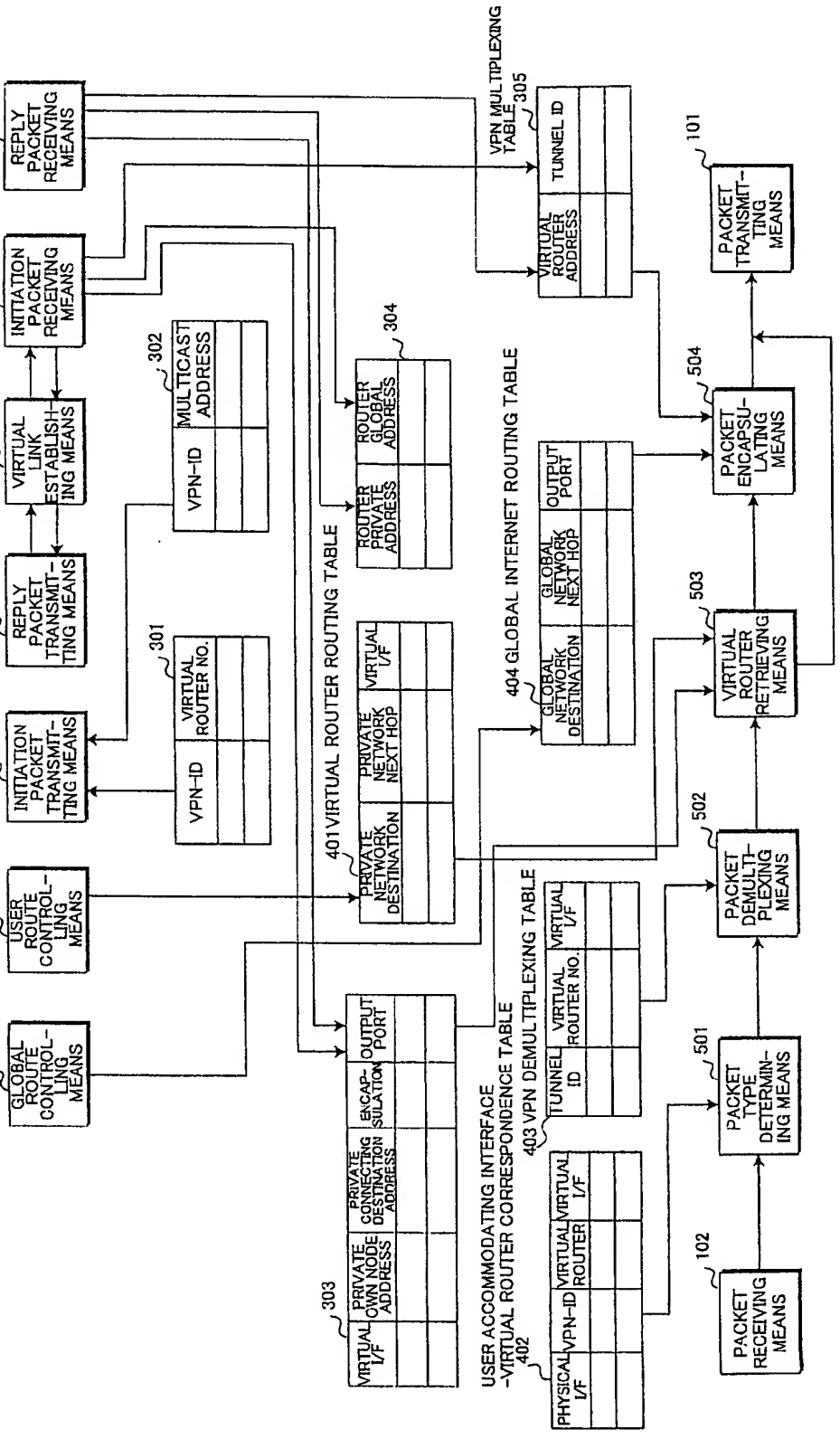

205 REPLY PACKET RECEIVING MEANS
204 INITIATION PACKET RECEIVING MEANS
203 VIRTUAL LINK ESTABLISHING MEANS
202 REPLY PACKET TRANSMITTING MEANS
201 INITIATION PACKET TRANSMITTING MEANS
506 USER ROUTE CONTROLLING MEANS
505 GLOBAL ROUTE CONTROLLING MEANS
VPN MULTIPLEXING TABLE 305
TUNNEL ID
VIRTUAL ROUTER ADDRESS
302 MULTICAST ADDRESS
VPN-ID
304
ROUTER GLOBAL ADDRESS
ROUTER PRIVATE ADDRESS
301
VIRTUAL ROUTER NO.
VPN-ID
401 VIRTUAL ROUTER ROUTING TABLE
PRIVATE NETWORK DESTINATION
PRIVATE NETWORK NEXT HOP
VIRTUAL I/F
404 GLOBAL INTERNET ROUTING TABLE
GLOBAL NETWORK DESTINATION
GLOBAL NETWORK NEXT HOP
OUTPUT PORT
303
VIRTUAL I/F
PRIVATE OWN NODE ADDRESS
PRIVATE CONNECTING DESTINATION ADDRESS
ENCAPSULATION
OUTPUT PORT
USER ACCOMMODATING INTERFACE -VIRTUAL ROUTER CORRESPONDENCE TABLE
402
PHYSICAL I/F
VPN-ID
VIRTUAL ROUTER
VIRTUAL I/F
403 VPN DEMULTIPLEXING TABLE
TUNNEL ID
VIRTUAL ROUTER NO.
VIRTUAL I/F
102 PACKET RECEIVING MEANS
501 PACKET TYPE DETERMINING MEANS
502 PACKET DEMULTIPLEXING MEANS
503 VIRTUAL ROUTER RETRIEVING MEANS
504 PACKET ENCAPSULATING MEANS
101 PACKET TRANSMITTING MEANS

FIG.14

DESTINATION	NEXT HOP	OUTPUT PORT
global1. 0/24	Direct	PR1-PP6
global2. 0/24	global1. 2	PR1-PP6
global3. 0/24	global1. 2	PR1-PP6
global4. 0/24	global1. 2	PR1-PP6

FIG.15

DESTINATION	NEXT HOP	OUTPUT VIRTUAL INTERFACE
private1. 6. 0/24	private1. 100. 3	V1-VR1-VP5
private1. 5. 0/24	private1. 100. 2	V1-VR1-VP6
private1. 2. 0/24	private1. 20. 2	V1-VR1-VP2

FIG.16

PHYSICAL INTERFACE	VPN-ID	VIRTUAL ROUTER	VIRTUAL INTERFACE
PR1-PP1	2	VPN2-VR1	V2-VR1-VP1
PR1-PP2	1	VPN1-VR1	V1-VR1-VP2

FIG.17

VIRTUAL INTERFACE	OWN ADDRESS	CONNECTING DESTINATION ADDRESS	ENCAP-SULATION	OUTPUT PORT
V1-VR1-VP2	private1.2.1	private1.20.11	NO	PR1-PP2
V1-VR1-VP5	private1.10.1	private1.100.3	YES	--
V1-VR1-VP6	private1.11.1	private1.100.2	YES	--

FIG.18

PRIVATE ADDRESS	GLOBAL ADDRESS
private1. 100. 2	global3. 2
private2. 100. 2	global3. 2
private1. 100. 3	global4. 2

FIG.19

CONNECTING DESTINATION VIRTUAL ROUTER ADDRESS	TRANSMITTING TUNNEL ID	TRANSMITTING SESSION ID
private1. 100. 2	300	202
private1. 100. 3	301	243
private2. 100. 2	1001	1201

FIG.20

RECEIVING TUNNEL ID	RECEIVING SESSION ID	RECEIVING VIRTUAL ROUTER	VIRTUAL INTERFACE
105	200	VPN1-VR1	V1-VR1-VP6
106	201	VPN1-VR1	V1-VR1-VP5
1102	1301	VPN2-VR1	V2-VR1-VP6

FIG.21

PRIOR ART

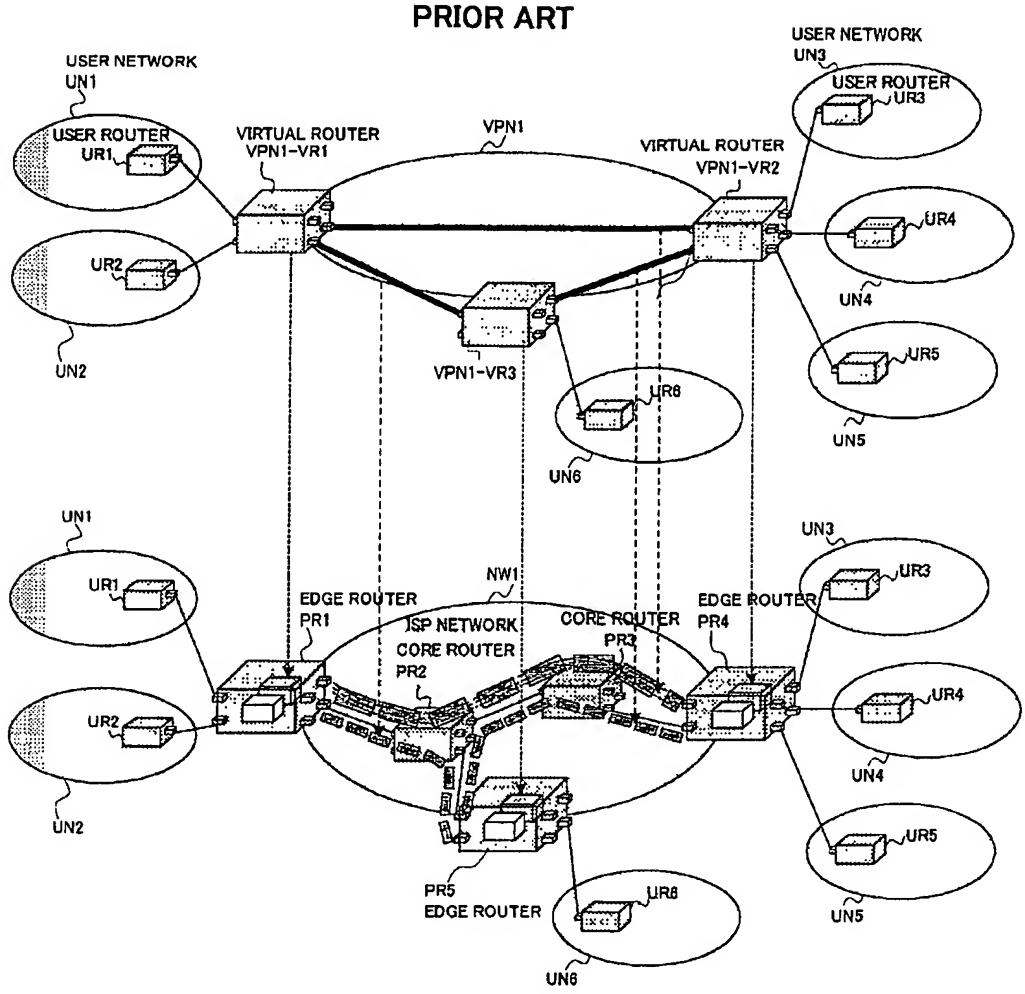


FIG.22

PRIOR ART

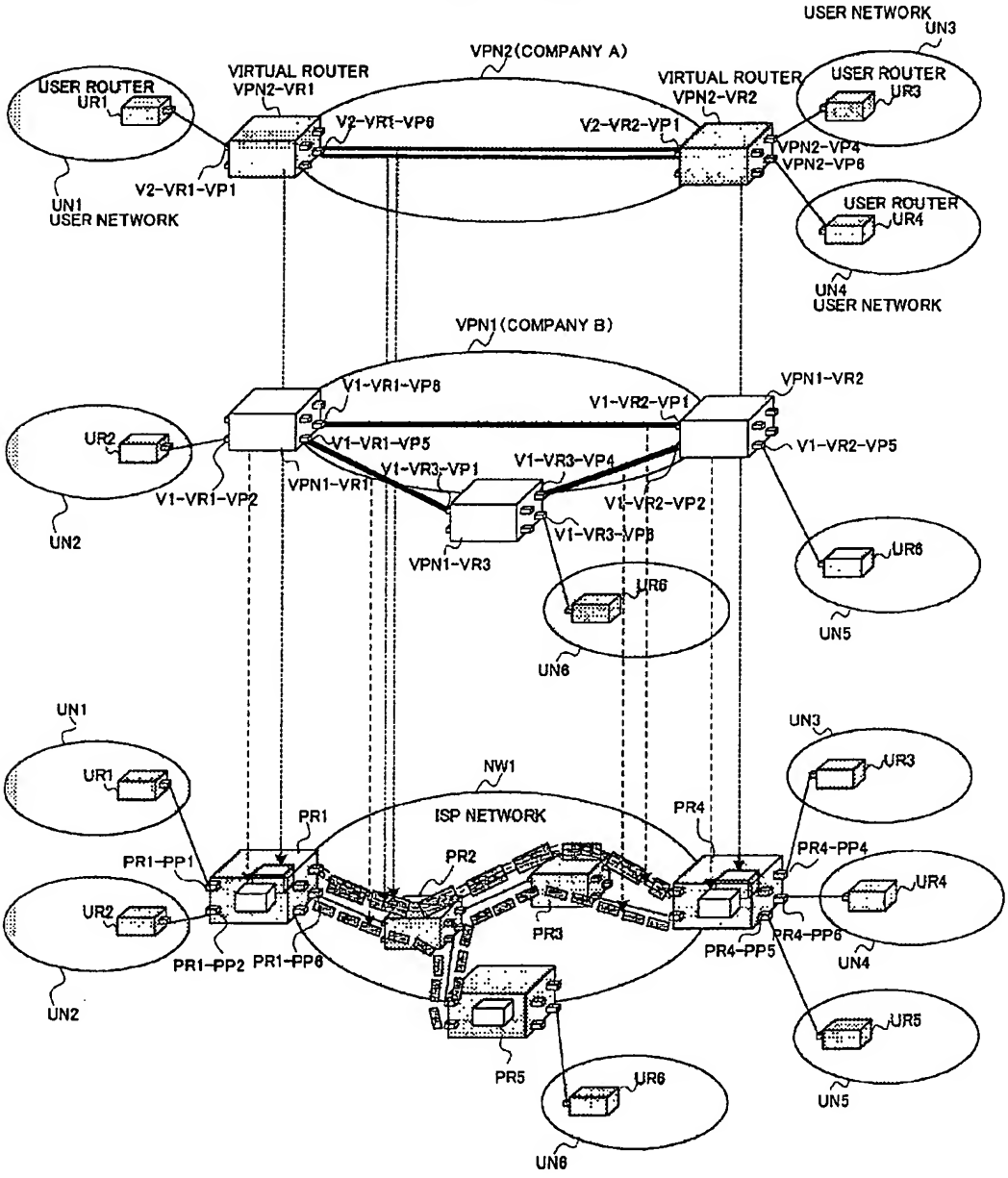


FIG.23

PRIOR ART

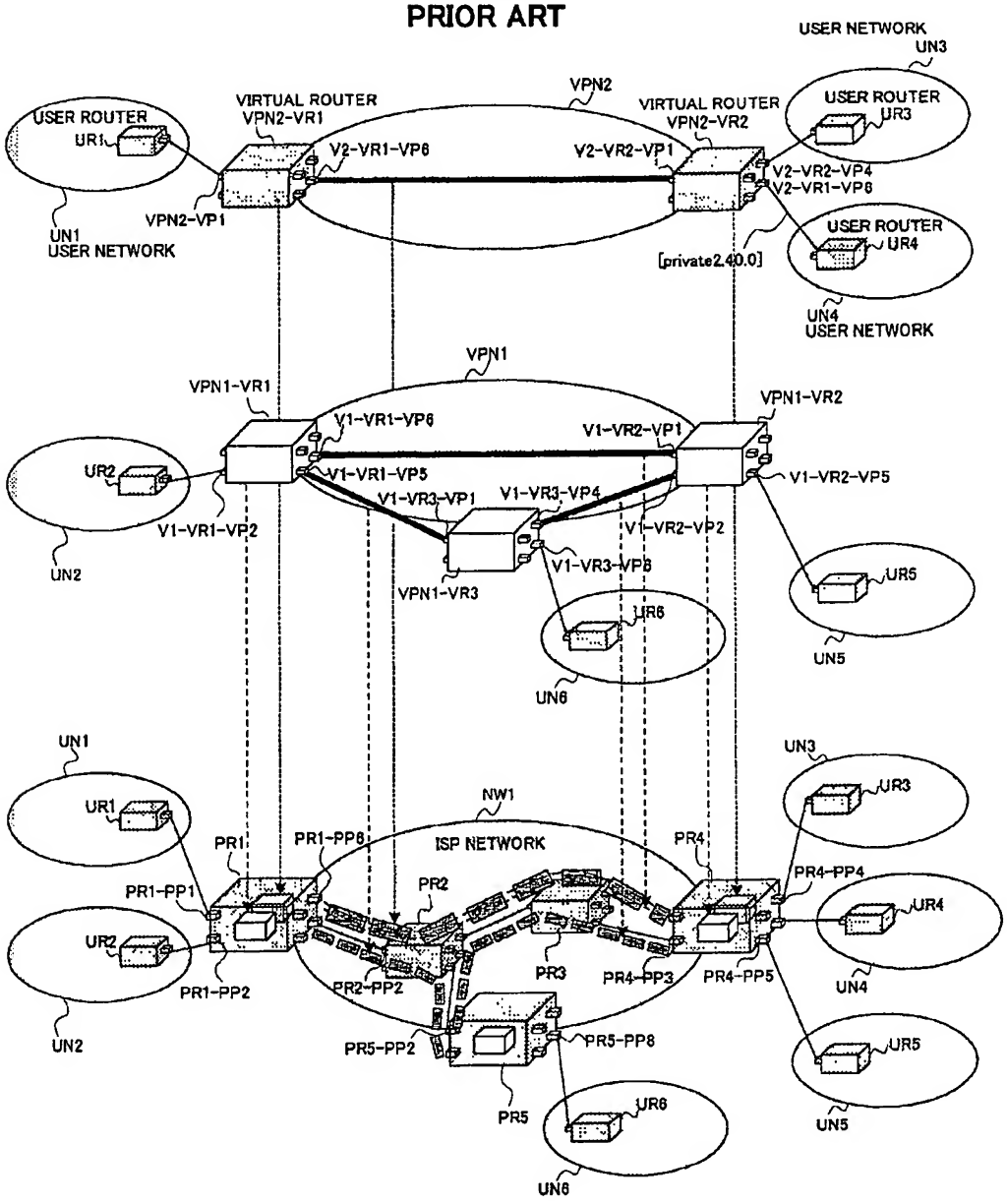


FIG.24
PRIOR ART

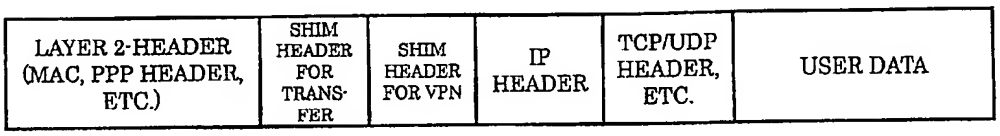


FIG.25
PRIOR ART

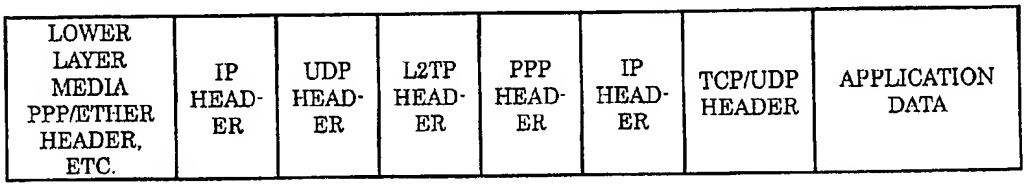


FIG.26
PRIOR ART

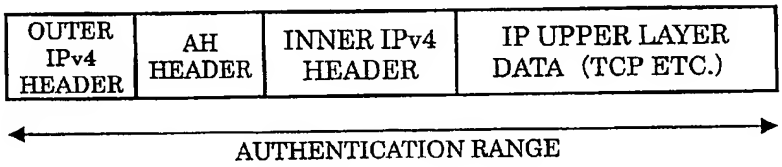
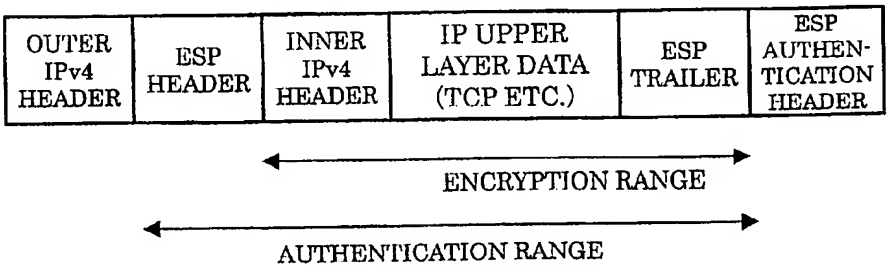


FIG.27
PRIOR ART



VIRTUAL NETWORK CONSTRUCTION METHOD, SYSTEM, AND RELAYING APPARATUS

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates to a virtual network construction method, a virtual network construction system, and a relaying apparatus, and in particular to a virtual network construction method, a virtual network construction system, and a relaying apparatus within a public data communication network.

[0003] 2. Description of the Related Art

[0004] Companies, enterprises, or the like having their sites which are referred to as user sites dispersed over a plurality of locations, have adopted various methods as inter-LAN connecting technology for connecting local area networks (LAN's) of the sites to construct intra-company networks or the like.

[0005] One of such methods is a leased line service connecting the user sites with leased lines, for example. However, since the leased line service is very expensive and its billing is proportional to the distance, the user company constructs the inter-LAN connection by connecting each site in line in order to economize the distance of the lines utilized as much as possible.

[0006] In this case, there has been a problem that when the communication is disabled at an intermediating user site due to a fault, the end to end communication is also disabled.

[0007] Thereafter, a virtual leased line service such as an ATM (Asynchronous Transfer Mode) service and an FR (Frame Relay) service which is less expensive compared to the leased line service appeared, so that the billing is performed in accordance with the number of virtual connections instead of the billing proportional to the distance.

[0008] As a result, network configuration connecting the LAN's of branch offices to a headquarter in the form of a star has increased and it has decreased that a fault at an intermediating site gives influences on the other sites.

[0009] Moreover, the spread of the Internet has enabled user companies to connect the dispersed user sites by using the Internet, which is a public data communication network, without using the virtual leased line service such as the ATM service and the FR service. Such a service is called an Internet VPN service, and the billing is performed by the number of physical sites connected. It is to be noted that VPN stands for Virtual Private Network.

[0010] Since the LAN of each user site (hereinafter, referred to as user network) generally uses private addresses in an Internet VPN service, packets cannot be flown unchanged into the Internet using global addresses.

[0011] Therefore, for the communications through the Internet (hereinafter, referred to as global Internet) between a plurality of user network sites, a so-called tunneling technique is required.

[0012] Namely, when transmitting a packet from the user network to the global Internet, a router connecting to the global Internet in the user network of the transmitting source encapsulates the packet to be transmitted from the user network with an IP packet having a global address and transmits it to a destination user network through the global Internet.

[0013] A router connecting to the global network in the destination user network decapsulates the packet after receiving it and then forwards it to a destination host computer within the destination user network.

[0014] In this case, each user network is required to be provided with a router connecting to the global network that is an apparatus capable of initiating and terminating a tunnel, i.e. encapsulating and decapsulating the packet. However, when processes become complicated the performance of that router declines, so that purchase of an expensive apparatus or upgrade is required in order to improve the performance.

[0015] Moreover, if there are numerous sites, various settings, such as routing information setting and logical interfaces setting, necessary for the connection to the global Internet become more complicated. In this case, the user company is required to educate managers for maintaining and managing the VPN, so that additional staffs and costs are required.

[0016] Consequently, a new VPN service has been devised in which the maintenance and management of the VPN are outsourced to a provider (Internet Service Provider; hereinafter abbreviated as ISP) or a carrier of the public data communication network so that the existing routers can be used in the user network without changes. Hereinafter, such a VPN service will be referred to as an IP-VPN (Internet Protocol-Virtual Private Network) service.

[0017] In the IP-VPN service, the tunnel initiating/terminating function is provided by a relaying apparatus within the public data communication network. Hereinafter, the relaying apparatus within the public data communication network having the tunnel initiating/terminating function will be occasionally referred to as an edge router. Moreover, in case there are a plurality of user sites and the user networks of the sites are connected to different routers, a routing control between the user networks is required wherein the edge router determines, for a packet transmitted from a user network, to which tunnel an encapsulated packet should be transmitted according to the destination user network. Such a routing control function is also provided by the edge router.

[0018] Namely, the edge router transfers the packet based on routing information of a private address of the user network, aside from the routing information of the global Internet.

[0019] In order to describe a general IP-VPN service, **FIG. 21** shows that virtual networks (hereinafter, referred to as private networks) constructed by tunnels connecting the user networks are overlaid to the global Internet operated by using global addresses, when the user networks are operated by using private addresses.

[0020] In **FIG. 21**, an ISP network NW1 providing a global address space has its backbone composed of edge routers PR1, PR4, and PR5, and core routers PR2 and PR3 which do not accommodate the user networks nor provide the tunnel initiating/terminating function within the public data communication network.

[0021] Now, a case will be considered where a user company desires to mutually connect user networks UN1-UN6 by using the IP-VPN service.

[0022] In this case, the user networks UN1-UN6 have existing routers (user routers) UR1-UR6 respectively, wherein the user routers UR1 and UR2 are connected to the edge router PR1, the user routers UR3-UR5 are connected to the edge router PR4, and the UR6 is connected to the edge router PR5, respectively.

[0023] In the edge routers PR1, PR4, and PR5, there are virtual routers VPN1-VR1-VPN1-VR3. Therefore, the user networks UN1-UN6 are connected to a virtual private network VPN1 which is a private address space through the virtual routers VPN1-VR1-VPN1-VR3 as shown extracted above the network NW1 in **FIG. 21**.

[0024] Conventionally proposed methods of such an IP-VPN service will be specifically described below.

[0025] (1)IETF RFC2547

[0026] Firstly, a method proposed as an IETF RFC2547 will be described referring to **FIG. 22**.

[0027] **FIG. 22** shows the same physical connection form as that of **FIG. 21**. However, in **FIG. 22**, different from **FIG. 21**, it is assumed that a user company (company A) having sites of the user networks UN1, UN3, and UN4 is different from a user company (company B) having sites of the user networks UN2, UN5, and UN6.

[0028] Therefore, in **FIG. 22**, a virtual private network VPN2 for the company A and a virtual private network VPN1 for the company B are separately constructed.

[0029] Also in **FIG. 22**, ports are shown as physical interfaces of the edge routers PR1, PR4, and PR5, e.g. ports PR1-PP1, PR1-PP2, and PR1-PP6 are shown in the edge router PR1.

[0030] Also, ports as virtual interfaces of virtual routers VPN1-VR1, VPN1-VR2, VPN1-VR3, VPN2-VR1, and VPN2-VR2 are shown, e.g. ports V2-VR1-VP1 and V2-VR1-VP6 are shown in the virtual router VPN2-VR1.

[0031] Hereinafter, the process of the IETF RFC2547 method will be described.

[0032] When the companies A and B respectively perform communications between their user networks, it is required that the packets are transferred through the ISP network NW1 in the virtual private networks VPN2 and VPN1 respectively.

[0033] The RFC2547 method realizes the VPN using a technique called a Multi Protocol Label Switching (MPLS) and a routing protocol called a Border Gateway Protocol.

[0034] The MPLS is a technique which enables a router on an IP route to replace an IP packet relaying process performed on a network layer with a label switching process performed on a datalink layer by using a label added to the packet, thereby reducing a process of route retrieval and relaying a packet at a high speed.

[0035] The label of MPLS assumes a value predetermined for an inter-router link between the routers sharing the links, so that upon receiving a packet with label, the router checks the label to determine where it should be relayed to, and adds a new label corresponding to the output link to the packet to be retransmitted.

[0036] A path in which the packet is transferred by the label is called a Label Switching Path (LSP). The LSP can be regarded as a tunnel in which the IP packet is encapsulated to be transferred by the label. Hereinafter, the LSP will be occasionally referred to as an MPLS tunnel.

[0037] Also, in the RFC2547 method, the routing protocol called the Border Gateway Protocol (hereinafter abbreviated as BGP) is used. In the edge routers, a routing control process realizing this protocol is activated, so that the routing control processes on the edge routers are connected in a full mesh. Alternatively, the edge routers can be connected starlike, so that they are connected through a route reflector providing an exchange function of a routing control packet similar to that provided in case of the full mesh connection.

[0038] In order to exchange the routing control packets by the MPLS tunnels between the edge routers connected in the full mesh, the LSP's are required to be pre-established so that the edge routers are connected in the full mesh. The LSP's established herein are realized by setting, in the routers, labels corresponding to inter-router links on routes for global IP prefixes of destination network. Such LSP's will be hereinafter referred to as level-1 tunnels. In the arrangement of **FIG. 22**, the level-1 tunnels are established between physical routers PR1-PR4, PR1-PR5, and PR4-PR5.

[0039] An administrator of ISP makes a port (I/F) number of the edge router correspond to a Route Distinguisher (hereinafter abbreviated as RD) as a user site identifier. In this case, the RD can be an arbitrary number which is unique for each user network managed by the provider network.

[0040] Also, there is another mapping between the VPN's and groups of RD's, that sets which user networks, distinguished by the RD's belong to the same VPN. By this mapping, e.g. the VPN2 and VPN1 are respectively made to correspond to the ports PR1-PP1 and PR1-PP2 of the edge router PR1. In the edge router, the VPN's are distinguished by VPN numbers, and the VPN numbers are used for managing the routing table independently per VPN, and for making user network accommodating ports correspond to the VPN.

[0041] Also, the administrator of ISP makes one-to-one correspondences between the port numbers and the virtual interfaces of the virtual routers for each port of the edge routers connected to the user networks.

[0042] By making such correspondences, e.g. the virtual interfaces V2-VR1-VP1 and V1-VR1-VP2 are respectively made to correspond to the ports PR1-PP1 and PR1-PP2 of the edge router PR1.

[0043] It is to be noted that the edge routers PR1, PR4, and PR5 have independent routing tables per VPN. These routing tables are generated by the routing control process (BGP) common to the VPN's and independently generated per virtual private network based on the routing information within all of the virtual private networks (VPN1 and VPN2 in case of **FIG. 22**) received from the local sites or remote sites.

[0044] At this time, the routing control process on the edge router assigns an RD to an address prefix of the received routing information from the user networks, so that the routing information can be distinguished per virtual private network.

[0045] Also, the edge routers have a function of searching through the routing table corresponding to the VPN by the port number of the port having received the data packet and of forwarding the packet received. This forwarding function has a virtual interface for transmitting the packet to the tunnel established between the edge routers.

[0046] The edge routers have different MPLS tunnels (level-2 tunnels) per destination prefix within the same VPN, so that different tunnels per destination can be identified.

[0047] The edge routers multiplex the tunnels for each prefix (level-2 tunnels), nested within the level-1 tunnel, between the edge routers. Actually, the edge routers doubly add the MPLS labels corresponding to the level-1 tunnel and the level-2 tunnel to the IP packet.

[0048] This can be seen in **FIG. 22**, where three level-2 tunnels are established in the level-1 tunnel between the edge routers PR1 and PR4. Namely, the three level-2 tunnels are the two tunnels between the virtual port V2-VR1-VP6 of the virtual router VPN2-VR1 and the virtual port V2-VR2-VP1 of the virtual router VPN2-VR2 established per address prefix, a single tunnel established between the virtual port V1-VR1-VP6 of the virtual router VPN1-VR1 and the virtual port V1-VR2-VP1 of the virtual router VPN1-VR2.

[0049] In the routing tables per VPN on the edge routers, a representing address of a next hop edge router and a virtual interface for transmission thereto for each destination prefix are written. The virtual interface is an entrance to the level-2 tunnel connected to the destination edge router.

[0050] In **FIG. 22**, the virtual interface V2-VR1-VP6 of the virtual router VPN2-VR1 within the edge router PR1 is the entrance to the level-2 tunnel connected to the destination edge router PR4.

[0051] The edge router assigns a different label for a level-2 tunnel per prefix, and adds a label for a level-1 tunnel determined by the representing address of the next hop edge router to transmit the packet to the physical port (PP) connected to the global Internet.

[0052] As to routing control process, the routing control process on each edge router generates independent routing tables per VPN by exchanging routing information both of the global Internet and of the VPN's through the level-1 tunnel established between the routers.

[0053] In the forwarding process, when the packet arrives at the physical port of the edge router from the user site, the edge router refers to the routing table corresponding to the VPN by the VPN number corresponding to the physical port which has received the packet and transmits the packet to the virtual interface connected to the next hop edge router.

[0054] When the virtual router transmits the packet to the virtual interface, practically, after the edge router adds a label (hereinafter, referred to as level-2 label) corresponding to the level-2 tunnel per prefix, the edge router adds a label (hereinafter, referred to as level-1 label) corresponding to the level-1 tunnel to the edge router on which the destination virtual router exists, and transmits the packet to the physical interface.

[0055] Also, when the edge router receives a packet with a label from the ISP network NW1, the next hop router and the output physical port are determined by the label, using a label table where a relaying operation is described. For example, in an MPLS implemented system by the Cisco Systems, Inc., in the United States, the level-1 label is removed at an LSR (label switching router) which is prior to the edge router by one hop, so that the edge router receives the packet with the level-2 label. The edge router checks the level-2 label, searches through the label table, and forwards the packet to the physical port connected to the user site. At this time, the level-2 label is removed from the packet to be forwarded.

[0056] (2)IETF draft draft-muthukrishnan-corevpn-arch-00.txt

[0057] Next, a method proposed as an IETF draft draft-muthukrishnan-corevpn-arch-00.txt will be described referring to **FIG. 23**.

[0058] The arrangement of **FIG. 23** is almost the same as that of **FIG. 22**. However, it is different in that there are two tunnels in the virtual private network VPN2 shown in **FIG. 22**, between the virtual interface V2-VR1-VP6 of the virtual router VPN2-VR1 and the virtual interface V2-VR2-VP1 of the virtual router VPN2-VR2, whereas only one tunnel is shown in **FIG. 23**.

[0059] This is because in this method, management per destination prefix is not performed.

[0060] Also, since the routing protocol between the virtual routers is not limited to the BGP in this method, tunnels are not always required to be established in the full mesh between the edge routers. However, establishing the tunnels in the full mesh is preferable considering that the end-to-end communication will be disturbed if a fault occurs in an edge router, and that the number of router hops of the relayed packet will be increased by relaying a number of edge routers.

[0061] In this case, the MPLS is used as the tunneling technique, and the administrator of ISP establishes the MPLS tunnel (level-1 tunnel) between every pair of edge routers in the same way as in the case of **FIG. 22**.

[0062] Also, different from **FIG. 22**, the edge router activates an independent virtual router per VPN, so that the same VPN-ID is set in the virtual routers belonging to the same VPN. The virtual routing function has the routing function for receiving the routing information within the user network and generating the routing table based on the received information, and the forwarding function for forwarding the received packet by searching through the routing table corresponding to the VPN-ID by the received port number. This forwarding function has the virtual interface for transmitting the packet to the tunnel established between the edge routers.

[0063] Also, the virtual routers on the edge routers having the same VPN-ID are connected with the virtual link on the global network. However, in order to make distinction from the traffics from the user sites having other VPN-ID's, the

virtual routers having other VPN-ID's use different virtual links (tunnels) per VPN (level-2 tunnel).

[0064] The edge router multiplexes the inter-virtual router links (level-2 tunnels) of the VPN's being nested within the level-1 tunnel between the edge routers. Practically, the edge router doubly adds the MPLS labels corresponding to the level-1 tunnel and the level-2 tunnel to the IP packet to be transmitted.

[0065] In order to determine which virtual router of the edge router is connected to the end of which level-2 tunnel, the virtual router on the edge router makes the label value of the level-2 tunnel correspond to a virtual I/F address of the destination virtual router which is the connecting destination of the tunnel in case an IP address is allocated to the virtual I/F or to the representative address of the destination virtual router in case of a point-to-point link wherein the IP address is not allocated to the virtual I/F.

[0066] Also, the administrator of ISP makes one-to-one correspondences between the virtual interfaces of the virtual routers and the port numbers of the ports connected to the user site.

[0067] The virtual routers having the same VPN-ID exchange the routing information of each other through the level-2 tunnel established between the edge routers, and then generate routing tables for that VPN-ID.

[0068] When the packet arrives at the physical port of the edge router from the user site, the edge router refers to the routing table corresponding to the VPN-ID by the VPN-ID corresponding to the physical port having received the packet and transmits the packet to the virtual interface connected to the next hop virtual router.

[0069] When the virtual router transmits the packet to the virtual interface, practically, after the edge router adds a label corresponding to the level-2 tunnel, the edge router adds the label corresponding to the level-1 tunnel to another edge router on which the destination virtual router exists, and transmits the packet to the physical interface.

[0070] When the edge router receives the packet with the label from the level-1 tunnel, the edge router checks the level-1 label of the encapsulated packet, determines whether the packet is addressed to itself to remove the label, or the packet should be forwarded by changing the label. If it is addressed to itself, the edge router checks the label corresponding to the level-2 tunnel and determines which virtual interface of the virtual router within the edge router should receive the packet. At this time, the edge router removes the level-2 label to pass the packet to the virtual interface.

[0071] The virtual router having received the packet at the virtual interface checks the destination address in an IP header of the IP packet, that is the destination address within the user network, forwards the packet to one of the virtual interfaces corresponding to the virtual ports connected to the user site by searching through the VPN routing table held by the virtual router.

[0072] It is to be noted that in the above-mentioned methods (1) and (2), the MPLS tunneling is used as the tunneling technique. In this case, the packet relayed by the MPLS tunnel has a format as shown in **FIG. 24** wherein SHIM headers are doubly added.

[0073] However, an L2TP (layer two tunneling protocol) tunnel and an IPsec (IP security protocol) tunnel are generally used as the IP tunnel which is a tunneling technique other than the MPLS tunnel.

[0074] A packet of the general L2TP tunnel has a format shown in **FIG. 25**. When the packet consisting of the IP header, a TCP/UDP header, and application data is transmitted through an L2TP tunnel, an L2TP header and a PPP header are added thereto associated with an encapsulation. Moreover, when the edge router transmits the encapsulated packet to the provider network, a lower layer media PPP/Ether header, and the like as well as the IP header and the UDP header are also added.

[0075] Also, in the general IPsec tunnel, there are cases where an AH (authentication header) having the authenticating function and where an ESP (encapsulating security payload) header having both functions of authentication and encryption. The formats of the respective packets relayed in the IPsec tunnel are shown in FIGS.**26** and **27**.

[0076] As shown in **FIG. 26**, in the packet using the AH header, an outer IPv4 header, the AH header, an inner IPv4 header, and IP upper layer data are objects of the authentication.

[0077] Also, as shown in **FIG. 27**, the packet using the ESP header is composed of the outer IPv4 header, the ESP header, the inner IPv4 header, the IP upper layer data, an ESP trailer, and an ESP authentication header. The range excluding the outer IPv4 header and the ESP authentication header therefrom is the object of the authentication. Moreover, the range further excluding the ESP header therefrom is the object of the encryption.

[0078] In order to provide the IP-VPN service, the administrator of ISP allocates the VPN numbers or the VPN-ID's to the ports of the edge routers connected to the user networks. In order to enable the communication between the sites belonging to the same VPN, it is required that the sites are mutually connected by the tunnels through the global network and that the communication should be distinguished from the communication between the sites having other VPN numbers or VPN-ID's.

[0079] In the IETF RFC2547 method, the edge routers are required to hold the relationship between the ports and the virtual private networks to which the ports belong, and to mutually connect the ports within the same virtual private network with the virtual links (level-2 tunnels).

[0080] In the RFC2547 method, the BGP session for connecting the BGP routing control process on the edge router is established by using the level-1 tunnel connecting the edge routers. The edge router multiplexes the routing information of all of the VPN's by using the BGP session to be exchanged. The edge router determines, based on the routing information, which ports accommodating the user sites should be connected with the layer 2 tunnel.

[0081] The edge router distributing the routing information by using the BGP protocol sets which routing information of which site belonging to which VPN should be distributed to which virtual router. Also, for the edge router having received the routing information by the BGP protocol, the administrator of ISP manually sets in the edge router that, in which virtual router the route received by the BGP

should be stored. Therefore, if the configuration of the VPN becomes complicated and the number of the VPN's increases, the setting becomes extremely complicated.

[0082] Generally, the BGP is the routing protocol mainly used by providers which is transit networks. There are not a few providers who realize the routing control by an OSPF (open shortest path first). Therefore, operating the BGP on all of the edge routers of the providers in order to realize the VPN has been a big hurdle.

[0083] On the other hand, in the method of draft-mush-ukrishnan-corevpn-arch-00.txt, the virtual routers belonging to the same VPN (having the same VPN-ID) are connected with the level-2 tunnels, so that the routing information received from a site belonging to a certain VPN is exchanged between the virtual routers using the level-2 tunnels which connect the virtual routers belonging to the VPN.

[0084] This method has been proposed based on the MPLS, and uses the Label Distribution Protocol (LDP) for establishing a Label Switching Path (LSP) which is the MPLS tunnel within the MPLS network, so that it cannot be applied to methods using the IP tunnel (L2TP,IPsec).

SUMMARY OF THE INVENTION

[0085] It is accordingly an object of the present invention, in a virtual network construction method, a virtual network construction system, and a relaying apparatus within a public data communication network, to find virtual routers on edge routers belonging to the same VPN, so that the virtual routers belonging to the same VPN can be mutually connected with tunnels (such as the L2TP tunnel or the IPsec tunnel) other than the LSP as well, in case routing information is exchanged between the virtual routers belonging to the same VPN as described in the draft-mushukrishnan-corevpn-arch-00.txt in order to realize the VPN without requiring complicated settings for controlling the routing information per VPN as in the RFC2547 method.

[0086] For the achievement of the above object, the virtual network construction method according to the present invention comprises steps of: generating and multicasting control packets each having set a multicast address predetermined per virtual network in first relaying apparatuses originating a virtual network within a public data communication network, and establishing virtual links to the first relaying apparatuses which are transmitting sources of the control packets upon receipt thereof and returning reply packets through the virtual links in second relaying apparatuses belonging to the multicast address group, whereby the virtual links are established between all pairs of the first and the second relaying apparatuses belonging to the multicast address group to construct the virtual network.

[0087] Namely, the first relaying apparatuses terminating the virtual communication network firstly generate the control packets each having set a multicast address predetermined per virtual network to be multicast to the address. Then, the second relaying apparatuses belonging to the multicast address group establish virtual links to the first relaying apparatuses which are the transmitting sources of the control packets triggered by receipt thereof, and return reply packets through the virtual links.

[0088] The first relaying apparatuses which are the transmitting sources of the control packets having received the returned reply packets are able to know between which relaying apparatuses the virtual links are established.

[0089] If such operations are performed by the first and the second relaying apparatuses, the virtual links are established between all of the first and the second relaying apparatuses belonging to the multicast address group, thereby enabling the construction of the virtual network.

[0090] Therefore, the relaying apparatuses terminating the virtual network within the public data communication network have only to hold the relationship between the virtual network and the multicast address, so that the management is simplified compared to the conventional RFC2547 method where the relationship between the ports and the virtual networks to which the ports belong is held to connect the ports belonging to the same virtual network with the virtual links.

[0091] Also, various conventional tunneling techniques can be used for the establishment of the virtual link, so that the tunneling technique is not limited to the MPLS tunneling technique as in the prior art.

[0092] In this case, the second relaying apparatuses may authenticate the control packets received.

[0093] Therefore, it becomes possible to avoid a problem which may be caused in association with the multicast control packet being received by someone not permitted to receive it.

[0094] The virtual links established by the virtual network construction method according to the present invention may comprise IP tunnels or MPLS tunnels.

[0095] Also, the virtual network construction system according to the present invention comprises: first relaying apparatuses for generating and multicasting, when starting a construction of a virtual network within a public data communication network, control packets each having set a multicast address predetermined per virtual network, and second relaying apparatuses for establishing virtual links to the first relaying apparatuses which are transmitting sources of the control packets upon receipt thereof and for returning reply packets through the virtual links, whereby the virtual links are established between all of the first and the second relaying apparatuses belonging to the multicast address group by operations thereof to construct the virtual network.

[0096] Namely, in the virtual network construction method according to the present invention, when starting a construction of a virtual network within a public data communication network, the first relaying apparatuses generate and multicast the control packets each having set a multicast address predetermined per virtual network.

[0097] The second relaying apparatuses having received the control packets are triggered by the reception thereof to establish the virtual links to the first relaying apparatuses which are transmitting sources of the control packets upon receipt thereof, and return the reply packets through the virtual link.

[0098] The first relaying apparatuses which are the transmitting sources of the control packets having received the returned reply packets are able to know between which relaying apparatuses the virtual links are established.

[0099] If such operations are performed by the first and the second relaying apparatuses, the virtual links are established between all of the first and the second relaying apparatuses belonging to the multicast address group, thereby enabling the construction of the virtual network.

[0100] Therefore, in this virtual network construction system, as in the virtual network construction method, the management is simplified compared to the conventional RFC2547 method and the tunneling technique is not limited to the MPLS tunneling technique.

[0101] In this case, the second relaying apparatuses establishing the virtual links may authenticate the control packets received.

[0102] Therefore, it becomes possible to avoid a problem which may be caused in association with the multicast control packet being received by someone not permitted to receive it.

[0103] The virtual links established by the virtual network construction system according to the present invention may comprise IP tunnels or MPLS tunnels.

[0104] Also, the relaying apparatus according to the present invention, which terminates a virtual network within a public data communication network comprises: means for generating and multicasting control packets each having set a multicast address predetermined per virtual network, and means for establishing virtual links to other relaying apparatuses which are transmitting sources of the control packets upon receipt thereof and for returning reply packets through the virtual links, whereby the virtual links are established between all of the relaying apparatuses belonging to the multicast address group to construct the virtual network.

[0105] Namely the relaying apparatus according to the present invention generates and multicasts the control packets each having set a multicast address predetermined per virtual network, establishes virtual links to the other relaying apparatuses which are the transmitting sources of the control packets upon receipt thereof, and returns the reply packets through the virtual links.

[0106] The relaying apparatuses which are the transmitting sources of the control packets having received the reply packets are able to know with which relaying apparatuses the virtual links are established.

[0107] When the relaying apparatuses terminating the virtual network within the public data communication network thus operate, the virtual links are established between all of the relaying apparatuses belonging to the multicast address, so that the virtual network can be constructed.

[0108] Therefore, by using this relaying apparatus, when constructing a virtual network within a public data communication network, the management is simplified compared to the conventional RFC2547 method and the tunneling technique is not limited to that of the MPLS tunnel.

[0109] Also, the relaying apparatus according to the present invention may further comprise means for authenticating the control packets received.

[0110] By using such means, it becomes possible to avoid a problem which may be caused in association with the multicast control packet being received by someone not permitted to receive it.

[0111] Moreover, the relaying apparatus according to the present invention may further comprise means for generating a routing table for each of a plurality of virtual networks logically independent of one another, and means for performing a packet relay of each private network based on the routing table.

[0112] Namely, routing tables are generated for a plurality of logically independent virtual networks, so that the packet relay of each virtual network is performed based on the routing information. Therefore, the packet relay in this case is performed by each virtual network logically independent of one another.

[0113] Thus, a logically independent packet relay in each virtual network is enabled without causing confusions between different virtual networks.

[0114] The virtual links established by the relaying apparatus according to the present invention may comprise IP tunnels or MPLS tunnels.

BRIEF DESCRIPTION OF THE DRAWINGS

[0115] **FIG. 1** is a network diagram for illustrating an embodiment of the present invention;

[0116] **FIG. 2** is a diagram showing an embodiment of IP addresses allocated to virtual interfaces shown in **FIG. 1**;

[0117] **FIG. 3** is a diagram showing an embodiment of IP addresses allocated to interfaces of edge routers shown in **FIG. 1**;

[0118] **FIG. 4** is a diagram showing an embodiment of IP addresses allocated to interfaces of user routers shown in **FIG. 1**;

[0119] **FIG. 5** is a block diagram for illustrating operations of the edge routers shown in **FIG. 1**;

[0120] **FIGS. 6A and 6B** are diagrams showing embodiments of VPN-ID—virtual router correspondence tables according to the present invention;

[0121] **FIG. 7** is a diagram showing an embodiment of a VPN-ID—multicast address correspondence table according to the present invention;

[0122] **FIG. 8** is a diagram showing a packet format of a tunnel initiation message according to the present invention;

[0123] **FIG. 9** is a diagram showing an embodiment of all field values of the tunnel initiation message shown in **FIG. 8**;

[0124] **FIG. 10A and 10B** are diagrams showing setting examples of tunnel ID's and session ID's according to the present invention;

[0125] **FIG. 11** is a diagram showing an embodiment of field values of a reply message according to the present invention;

[0126] **FIG. 12** is a block diagram showing a connection example of means which can be added to the arrangement of **FIG. 5**;

[0127] **FIG. 13** is a block diagram showing a detailed function of an edge router shown in **FIG. 1**;

[0128] **FIG. 14** is a diagram showing an embodiment of a global Internet routing table according to the present invention;

[0129] **FIG. 15** is a diagram showing an embodiment of a virtual router routing table according to the present intention;

[0130] **FIG. 16** is a diagram showing an embodiment of a correspondence table of interface which accommodates user network to virtual router according to the present invention;

[0131] **FIG. 17** is a diagram showing an embodiment of a virtual interface management table according to the present invention;

[0132] **FIG. 18** is a diagram showing an embodiment of a private address resolution table according to the present invention;

[0133] **FIG. 19** is a diagram showing an embodiment of a VPN multiplexing table according to the present invention;

[0134] **FIG. 20** is a diagram showing an embodiment of a VPN demultiplexing table according to the present invention;

[0135] **FIG. 21** is a diagram showing general overlays of VPN's on a global Internet;

[0136] **FIG. 22** is a network diagram showing a prior art VPN arrangement (1);

[0137] **FIG. 23** is a network diagram showing a prior art VPN arrangement (2);

[0138] **FIG. 24** is a diagram showing a packet format within an MPLS tunnel in the prior art VPN arrangements (1) and (2);

[0139] **FIG. 25** is a diagram showing a packet format within a general L2TP tunnel;

[0140] **FIG. 26** is a diagram showing a packet format within a general IPsec tunnel using an AH header; and

[0141] **FIG. 27** is a diagram showing a packet format within a general IPsec tunnel using an ESP header.

[0142] Throughout the figures, like reference numerals indicate like or corresponding components.

DESCRIPTION OF THE EMBODIMENTS

[0143] An embodiment of the present invention will be described referring to **FIG. 1**. This embodiment has the same arrangement as that of **FIG. 23**, except that a host having an IP address [private1.2.23] is connected to a user network **UN2**, and a server having an IP address [private1.5.25] is connected to a user network **UN5**.

[0144] Also, in order to describe specifically, physical ports, virtual interfaces, networks, and tunnels have their corresponding IP addresses indicated by square brackets [].

[0145] Interfaces beginning with "lo0" are called loop back interfaces, and are not connected to any of the physical/logical links. Addresses of these interfaces are often used as ones representing the routers.

[0146] It is to be noted that an IP address in IPv4 is denoted as a four-byte integer delimited per byte such as "168.254.192.0". However, the IP address is denoted by substituting the upper two bytes or three bytes with a character string such as "private1" or "global" in the present embodiment.

[0147] Also, as an address notation, an IP address ending with "/24" indicates that there are 24 masking bits, and is mainly used to indicate a bit length of a network IP address within the IP address.

[0148] It is to be noted that **FIG. 2** shows IP addresses assigned to virtual interfaces of virtual routers within the VPN1 shown in **FIG. 1**. For example, an IP address [private1.20.1] is made to correspond to a virtual interface V1-VR1-VP2.

[0149] Also, **FIG. 3** shows IP addresses assigned to interfaces of the edge routers PR1, PR4, and PR5 shown in **FIG. 1**. For example, an IP address [private2.10.1] is made to correspond to the interface PR1-PP1.

[0150] Moreover, **FIG. 4** shows IP addresses assigned to interfaces of the user routers UR1-UR6 shown in **FIG. 1**. For example, an IP address [private2.10.2] is made to correspond to the UR1-PP1.

[0151] Firstly, a virtual network construction procedures will be described taking the procedures of the edge routers PR1 and PR4 as an example.

[0152] **FIG. 5** shows internal arrangements of the edge routers PR1 and PR4 within the ISP network NW1 shown in **FIG. 1**. However, for the sake of description, the edge routers PR1 and PR4 are shown outside of the ISP network NW1.

[0153] Both of the edge routers PR1 and PR4 have the same arrangement, and have packet transmitting means **101**, packet receiving means **102**, initiation packet transmitting means **201**, reply packet transmitting means **202**, virtual link establishing means **203**, initiation packet receiving means **204**, and reply packet receiving means **205**.

[0154] Moreover, both of the edge routers also have same tables, which include a VPN-ID—virtual router correspondence table **301**, a VPN-ID —multicast address correspondence table **302**, a virtual interface management table **303**, a private address resolution table **304**, and a VPN multiplexing table **305**.

[0155] It is to be noted that VPN-ID's per VPN and corresponding multicast addresses thereof are predetermined to be common within the network NW1 by an administrator of ISP.

[0156] Assuming that the VPN-ID's of VPN1 and VPN2 shown in **FIG. 1** are respectively "1" and "2", the VPN-ID—virtual router correspondence table **301** within the edge router PR1 shown in **FIG. 5** is set as shown in **FIG. 6A**, so that e.g. VPN1-VR1 is made to correspond to VPN-ID=1. Similarly, **FIG. 6B** shows an example of the VPN-ID—virtual router correspondence table **301** within the edge router PR4.

[0157] Also, since the VPN-ID—multicast address correspondence table **302** is common within the network NW1, the edge routers PR1 and PR4 have the same contents. **FIG. 7** shows an example of the VPN-ID—multicast address correspondence table **302**, wherein a multicast address [239.192.0.1] is set to correspond to VPN-ID=1.

[0158] Moreover, it is assumed that all of the routers PR1-PR5 within the network NW1 shown in **FIG. 1** have a multicast routing protocol activated in the global address space, so that they are capable of distributing multicast packets.

[0159] As for the tunneling technique, protocols such as the L2TP and IPsec can be used. In this embodiment, procedures for automatically establishing tunnels between virtual routers belonging to the same VPN by the edge routers PR1 and PR4 in case the L2TP is used as the tunneling technique will be described.

[0160] (1) In **FIG. 5**, the initiation packet transmitting means **201** of the edge router PR1 firstly generates tunnel initiation messages (hereinafter, referred to as initiation messages), which are control packets, referring to the VPN-ID—virtual router correspondence table **301** per VPN-ID set therein, and refers to the VPN-ID—multicast address correspondence table **302** to transmit the initiation messages having set the multicast addresses corresponding to the VPN-ID's as destination addresses through the packet transmitting means **101** to the network NW1.

[0161] A packet format of the initiation message is shown in **FIG. 8**, and field values of the initiation message corresponding to VPN-ID=1, for example, are shown in **FIG. 9**.

[0162] (2) The edge router PR4, when the initiation packet receiving means **204** receives the above-mentioned initiation message through the packet receiving means **102**, uses the virtual link establishing means **203** to establish a tunnel to a SRC IP address, the IP address of the edge router PR1 in this case, in the initiation message.

[0163] At this time, since a "tunnel type" field value in the initiation message indicates "0", that means the L2TP tunnel, the edge router PR4 establishes the L2TP tunnel.

[0164] By exchanging L2TP protocol packets, the edge router PR4 obtains a tunnel ID and a session ID of the L2TP tunnel. In case of an L2TP tunnel, when the direction of reply from the edge router receiving the initiation message is named an upward direction, and the opposite direction is named a downward direction, tunnels in both directions are established simultaneously.

[0165] Therefore, an upward tunnel (PR4→PR1) and a downward tunnel (PR1→PR4), respectively, can have values of the tunnel ID and the session ID as shown in **FIGS. 10A and 10B**, for example.

[0166] Next, the virtual router VPN1-VR2 corresponding to the VPN-ID in the message generates a new virtual interface (see V1-VR2-VP1 in **FIG. 1**), and the correspondence between the virtual interface V1-VR2-VP1 and the connecting destination address (SRC IP of the initiation message) are entered into the virtual interface management table **303**.

[0167] Also, correspondences between the generated virtual interface V1-VR2-VP1 and the upward tunnel's tunnel ID and session ID are entered into the VPN multiplexing table **305**.

[0168] Moreover, a correspondence between the IP address of the edge router PR1 that is the transmitting source of the initiation message included in the IP header of the initiation message (IP address of PR1-PP6 in this case) and the IP address of the transmitting source virtual router VPN1-VR1 included in the SRC IP field of the initiation message is entered in the private address resolution table **304**.

[0169] (3) Then, the edge router PR4 transmits a reply message through the established tunnel. The packet format of the reply message is the same as that of the initiation message shown in **FIG. 8** and the field values of the reply message are as shown in **FIG. 11**.

[0170] (4) The edge router PR1 having received the above-mentioned reply message generates a new virtual interface V1-VR1-VP6 to the VPN1-VR1 corresponding to the VPN-ID within the reply message.

[0171] Thereafter, reply packet receiving means set the correspondences among the virtual router, the virtual interface, and the tunnel ID and the session ID within the reply message in a demultiplexing table as later described. The demultiplexing table is referred by the edge router PR1 having received the reply packet through the L2TP tunnel to determine which virtual interface of which virtual router should receive the packet according to the values of the session ID and the tunnel ID.

[0172] The above-mentioned procedures (1)-(4) are performed in the same way in case the initiation message is transmitted from the edge router PR4.

[0173] In the foregoing description referring to **FIG. 5**, the procedures between the two edge routers PR1 and PR4 have been described. However, there are actually a number of edge routers, so that if there are three edge routers as shown in **FIG. 1**, for example, the initiation messages multicast by the edge router PR1 in connection with the VPN1 are received by the edge routers PR4 and PR5 belonging to the multicast address group of the VPN1, and those in connection with the VPN2 are received by only the edge router PR4 belonging to the multicast address group of the VPN2.

[0174] When such operations are mutually performed by all of the edge routers within the network NW1, L2TP tunnels can be established in the full mesh between virtual routers included in the same VPN for a plurality of VPN's.

[0175] It is to be noted that **FIG. 12** shows an example of connecting control packet authenticating means **103** and authentication database **104** to the initiation packet receiving means **204** and the reply packet receiving means **205** in case the control packet authenticating means **103** and the authentication database **104** are provided in the edge routers PR1 and PR4 shown in **FIG. 5**.

[0176] In this case, a password common to the edge routers managed by the provider is entered into the authentication database **104** of the edge routers.

[0177] As for the operation, the initiation packet receiving means **204** receives the initiation message from the edge router PR1 only when the control packet authenticating means **103** authenticate the password in the received initiation packet as the password entered into the authentication database **104**.

[0178] Also, the reply packet receiving means **205** receives the reply packet only when the control packet authenticating means **103** authenticate the password in the received reply packet as the password entered into the authentication database **104**.

[0179] In the VPN thus constructed by establishing the tunnels in the full mesh, the actual packet relaying process performed by the edge routers will be described hereinafter.

[0180] Communications within a provider network realizing the VPN can be separately considered in the following two stages:

[0181] (1) Backbone network communications

[0182] (2) Overlay network communications

[0183] The backbone network communications (1) are communications using global addresses realized by (physical) routers having the internet routing information within the provider network and by physical/logical links connecting the routers based on the Internet routing information managed by the provider network.

[0184] The overlay network communications (2) are communications using private addresses realized by virtual routers managing the user routing information and by tunnels connecting the virtual routers generated virtually on the backbone network based on the intranet routing information held by users. The overlay network communications are realized by encapsulating overlay network packets as communication packets of the backbone network to be transferred over the backbone network.

[0185] In order to describe the packet relaying processes performed by the edge routers realizing such backbone network communications (1) and overlay network communications (2), **FIG. 13** shows a more detailed arrangement of the edge router common to the edge routers PR1 and PR4 shown in **FIG. 5**.

[0186] In addition to the arrangement of the edge router shown in **FIG. 5**, the embodiment of **FIG. 13** is provided with packet type determining means **501**, packet demultiplexing means **502**, virtual router retrieving means **503**, packet encapsulating means **504**, global route controlling means **505**, and user route controlling means **506** as packet relaying process means.

[0187] Moreover, a virtual routing table **401**, a user accommodating interface—virtual router correspondence table **402**, a demultiplexing table **403**, and a global Internet routing table **404** are shown therein.

[0188] Hereinafter, the packet relaying process procedure of the edge router PR1 shown in **FIG. 1** will be described assuming that the edge router PR1 has the arrangement shown in **FIG. 13**. It is to be noted that in the arrangement of **FIG. 13**, as in the case of **FIG. 5**, the control packet authenticating means **103** and the authentication database **104** may be provided as shown in **FIG. 12**, while the description is omitted hereinafter.

[0189] The global route controlling means **505** exchanges global address routing information with the global route controlling means **505** on the other router in the Internet to generate the global Internet routing table **404**.

[0190] **FIG. 14** shows an example of the global Internet routing table **404** of the edge router PR1. As shown in **FIG. 1**, an address [global1.0/24] is an IP address allocated to a network connecting the edge router PR and the core router PR2.

[0191] Therefore, in **FIG. 14**, the address [global1.0/24] is made to correspond to "next HOP"="direct" and "output port"=PR1-PP6.

[0192] Also, the user route controlling means **506** of the edge router PR1 exchanges user routing information shown by private addresses, with other user route controlling means **506** of edge routers PR4 and PR5 on the network NW1, or with the user route controlling means on the user routers UR1 and UR2 in the user networks UN1 and UN2 to generate the virtual router routing table **401** per virtual router.

[0193] **FIG. 15** shows a virtual router routing table of the virtual router VPN1-VR1 as an example of the virtual router routing table **401** of the edge router PR1. For example, a route having a destination address [private1.6.0/24] is made to correspond to "next HOP"=[private1.100.3] and "output virtual I/F"=V1-VR1-VP5 as shown in **FIG. 15**.

[0194] This indicates that the route from the virtual router VPN1-VR1 shown in **FIG. 1** to the user network UR6 having an address [private1.6.0/24] passes through the virtual router VPN1-VR3 having an address [private1.100.3] and that the output virtual I/F in this case is V1-VR1-VP5.

[0195] **FIG. 16** shows an example of the user accommodating interface—virtual router correspondence table **402** of the edge router PR1 in this case.

[0196] For example, VPN-ID=2, virtual router=VPN2-VR1, and virtual interface V2-VR1-VP1 are made to correspond to the physical interface PR1-PP1.

[0197] Also, **FIG. 17** shows an example of the virtual interface management table **303** of VPN1-VR1. In this case, e.g. its own address [private1.2.1], a connecting destination address [private1.20.11], encapsulation ="NO", and the output port PR1-PP2 are made to correspond to the virtual interface V1-VR1-VP2. The encapsulation field indicates whether or not the encapsulation should be performed. In this case, the encapsulation is not performed since the user router UR2 of the user network UN2 is connected to the virtual interface V1-VR1-VP2.

[0198] In **FIG. 17**, e.g. the virtual interface V1-VR1-VP5 is connected to the virtual router VPN1-VR3 having an address [private1.100.3] through the L2TP tunnel, so that the encapsulation field denotes "YES".

[0199] Also, **FIG. 18** shows an example of the private address resolution table **304**. The private address resolution table is a table for obtaining, by the IP address of the destination virtual router, a global IP address of the edge router where the destination virtual router is located. When the virtual router transmits a packet to the next hop virtual router, it actually encapsulates the packet by a new packet header having a global address, so that a global IP address of the edge router is required for transmission to the edge router. In this case, e.g. a private address [private1.100.2] is made to correspond to a global address [global3.21].

[0200] Moreover, **FIG. 19** shows an example of the VPN multiplexing table **305**. The VPN multiplexing table is a table describing to which IP tunnel a packet should be transmitted, based on the address, upon transmission thereof to the destination virtual router. In this case, e.g. a transmitting tunnel ID=300 and a transmitting session ID=202

are made to correspond to a connecting destination virtual router address [private1.100.2].

[0201] It is now assumed that the host [private1.2.231] within the user network UN2 having an address [private2.0/24] shown in **FIG. 1** accesses the server [private1.5.25] within the user network UN5 having an address [private1.5.0/241].

[0202] When the packet from the user network UN2 arrives at the port PR1-PP2 of the edge router PR1, the edge router PR1 refers to the user accommodating interface—virtual router correspondence table **402** (see **FIG. 16**) by the port number (PR1-PP2) having received the packet, identifies VPN-ID=1 and the virtual router VPN1-VR1 of the VPN to which the user network UN2 belongs, and passes the received packet to the virtual router VPN1-VR1.

[0203] The virtual router VPN1-VR1 having received the packet refers to the virtual router routing table **401** (see **FIG. 15**) including the routing information of the user networks belonging to VPN1, and obtains a next HOP address [private1.100.2] of the next HOP virtual router VPN1-VR2 and the virtual interface V1-VR1-VP6 that is the output virtual I/F made to correspond to the destination user network [private1.5.0/24].

[0204] The virtual router VPN1-VRL transmits the packet to the virtual interface V1-VR1-VP6 to which the next HOP virtual router VPN1-VR2 is connected. At this time, the edge router PR1 refers to the VPN multiplexing table **305** (see **FIG. 19**) to encapsulate the packet by the L2TP.

[0205] In this example, the entry of [private1.100.2] in the VPN multiplexing table **305** matches, so that the transmitting tunnel ID=300 and the transmitting session ID=202 are obtained.

[0206] Also, the private address resolution table **304** (see **FIG. 18**) is searched through to determine the global address [global3.2] of the next HOP edge router PR4 by the address [private1.100.2] of the next HOP virtual router VPN1-VR2.

[0207] The edge router PR1 encapsulates the packet received from the user network UN2 by the L2TP, and adds an IP header having a destination IP address of the previously obtained global address [global3.2], and then searches through the global Internet routing table **404** (see **FIG. 14**) to transmit the encapsulated packet to the interface PR1-PP6 shown in the output port field.

[0208] Conversely, the operations in case a reply is returned to the host [private1.2.23] from the server [private1.5.25] will be described below.

[0209] The edge router PR1, upon receiving the reply packet encapsulated by the L2TP from the physical interface PR1-PP6, refers to the VPN demultiplexing table **403** using the tunnel ID and the session ID within the encapsulated header as keys.

[0210] **FIG. 20** shows an example of the VPN demultiplexing table **403** of the edge router PR1 in this case. This table is referred in order to determine which virtual interface of which virtual router should receive the packet based on the values of the session ID and the tunnel ID when the edge router PR1 receives the packet from the L2TP tunnel.

[0211] For example, as shown in **FIG. 20**, when a packet having the receiving tunnel ID=105 and the receiving session ID=200 is received, it is seen that the packet is received at the virtual interface V1-VR1-VP6 of the virtual router VPN1-VR1.

[0212] At this time, the edge router PR1 removes the encapsulated header to pass the received packet to the virtual router VPN1-VR1. The virtual router VPN1-VR1 having received the packet at the virtual interface V1-VR1-VP6 checks the destination address, which is the destination address within the user network, in the IP header of the received IP packet, having a private address after removal of the L2TP header, and searches through the virtual router routing table **401** (see **FIG. 15**) of the virtual router VPN1-VR1.

[0213] In this case, an entry with the destination address [private1.2.0/24] is hit, so that it is seen that the packet can be transmitted to the virtual interface V1-VR1-VP2.

[0214] Therefore, the edge router PR1 refers to the virtual interface management table **303** (see **FIG. 17**) to transmit the packet to the output port PR1-PP2 made to correspond to the virtual interface V1-VR1-VP2. At this time, since the encapsulation field of the table **303** indicates "NO", the encapsulation is not performed.

[0215] It is to be noted that the present embodiment has dealt with the case where the L2TP tunneling is used as a tunneling technique. The format of the encapsulated packet transmitted through the L2TP tunnel in such a case is the same as that shown in **FIG. 25**.

[0216] However, since the tunneling techniques are not limited in the present invention, an IPsec tunnel or an MPLS tunnel can also be applied.

[0217] As described above, a virtual network construction method, a virtual network construction system, and a relaying apparatus according to the present invention are so arranged that control packets each having set a multicast address are multicast, and upon reception of the control packets by the relaying apparatuses belonging to the multicast address group, virtual links to the transmitting sources of the control packets are established by the received relaying apparatus and reply packets are returned through the virtual links, whereby the virtual links are established between all of the relaying apparatuses belonging to the multicast address group to establish the virtual network. Therefore, complicated VPN management becomes unnecessary and various tunneling techniques become available.

What we claim is:

1. A virtual network construction method comprising the steps of:

generating and multicasting control packets each having set a multicast address predetermined per virtual network in first relaying apparatuses originating a virtual network within a public data communication network, and

establishing virtual links to the first relaying apparatuses which are transmitting sources of the control packets upon receipt thereof and returning reply packets through the virtual links in second relaying apparatuses belonging to the multicast address group,

whereby the virtual links are established between all pairs of the first and the second relaying apparatuses belonging to the multicast address group to construct the virtual network.

2. The virtual network construction method as claimed in claim 1 wherein the second relaying apparatuses authenticate the control packets received.

3. The virtual network construction method as claimed in claim 1 wherein the virtual links comprise IP tunnels.

4. The virtual network construction method as claimed in claim 1 wherein the virtual links comprise MPLS tunnels.

5. A virtual network construction system comprising:

first relaying apparatuses for generating and multicasting, when starting a construction of a virtual network within a public data communication network, control packets each having set a multicast address predetermined per virtual network, and

second relaying apparatuses for establishing virtual links to the first relaying apparatuses which are transmitting sources of the control packets upon receipt thereof and for returning reply packets through the virtual links,

whereby the virtual links are established between all of the first and the second relaying apparatuses belonging to the multicast address group by operations thereof to construct the virtual network.

6. The virtual network construction system as claimed in claim 5 wherein the second relaying apparatuses establishing the virtual links authenticate the control packets received.

7. The virtual network construction system as claimed in claim 5 wherein the virtual links comprise IP tunnels.

8. The virtual network construction system as claimed in claim 5 wherein the virtual links comprise MPLS tunnels.

9. A relaying apparatus, which terminates a virtual network within a public data communication network comprising:

means for generating and multicasting control packets each having set a multicast address predetermined per virtual network, and

means for establishing virtual links to other relaying apparatuses which are transmitting sources of the control packets upon receipt thereof and for returning reply packets through the virtual links,

whereby the virtual links are established between all of the relaying apparatuses belonging to the multicast address group to construct the virtual network.

10. The relaying apparatus as claimed in claim 9, further comprising means for authenticating the control packets received.

11. The relaying apparatus as claimed in claim 9, further comprising means for generating a routing table for each of a plurality of virtual networks logically independent of one another, and means for performing a packet relay of each virtual network based on the routing table.

12. The relaying apparatus as claimed in claim 9 wherein the virtual links comprise IP tunnels.

13. The relaying apparatus as claimed in claim 9 wherein the virtual links comprise MPLS tunnels.

* * * * *

US 20020085498A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0085498 A1

Nakamichi et al. (43) Pub. Date: Jul. 4, 2002

(54) DEVICE AND METHOD FOR COLLECTING TRAFFIC INFORMATION

(76) Inventors: Koji Nakamichi, Kawasaki (JP); Kenya Takashima, Kawasaki (JP); Toshio Soumiya, Kawasaki (JP)

Correspondence Address:
KATTEN MUCHIN ZAVIS ROSENMAN
575 MADISON AVENUE
NEW YORK, NY 10022-2585 (US)

(21) Appl. No.: 09/981,019

(22) Filed: Oct. 16, 2001

(30) Foreign Application Priority Data

Dec. 28, 2000 (JP) 2000-400634

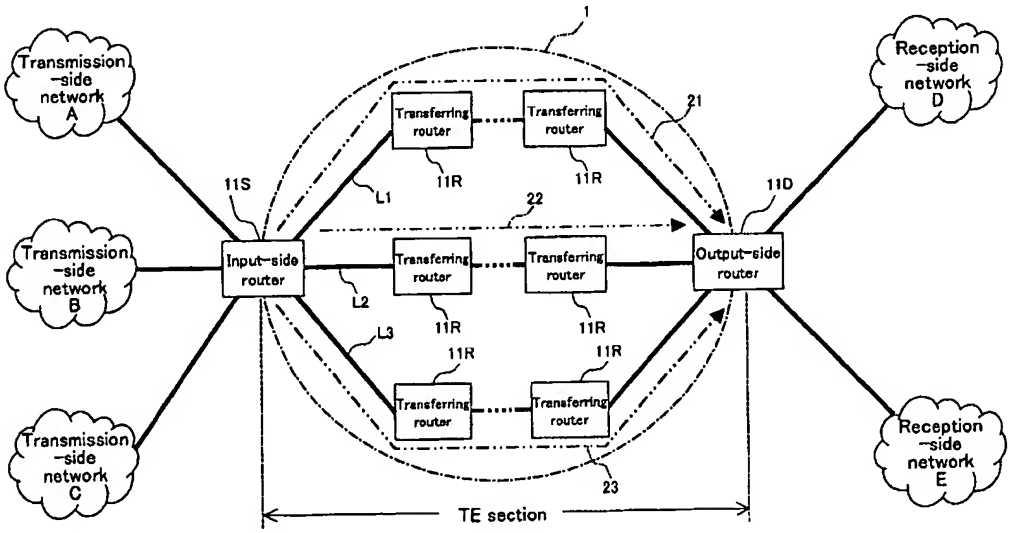
Publication Classification

(51) Int. Cl.[7] H04L 12/26

(52) U.S. Cl. 370/236; 370/400

(57) ABSTRACT

The present invention provides a traffic information collecting device and method. A node in a communication network collects a first traffic information of a first link connected to an own node. And also the node receives second traffic information of second links connected to the other nodes. The second traffic information is transmitted from the other nodes. The node stores the first and second information. In consequence, the node can be informed of the conditions of the traffic of each link in the communication network based on the stored information. As the result, it becomes possible to control the load sharing in correspondence to the conditions of the traffic.



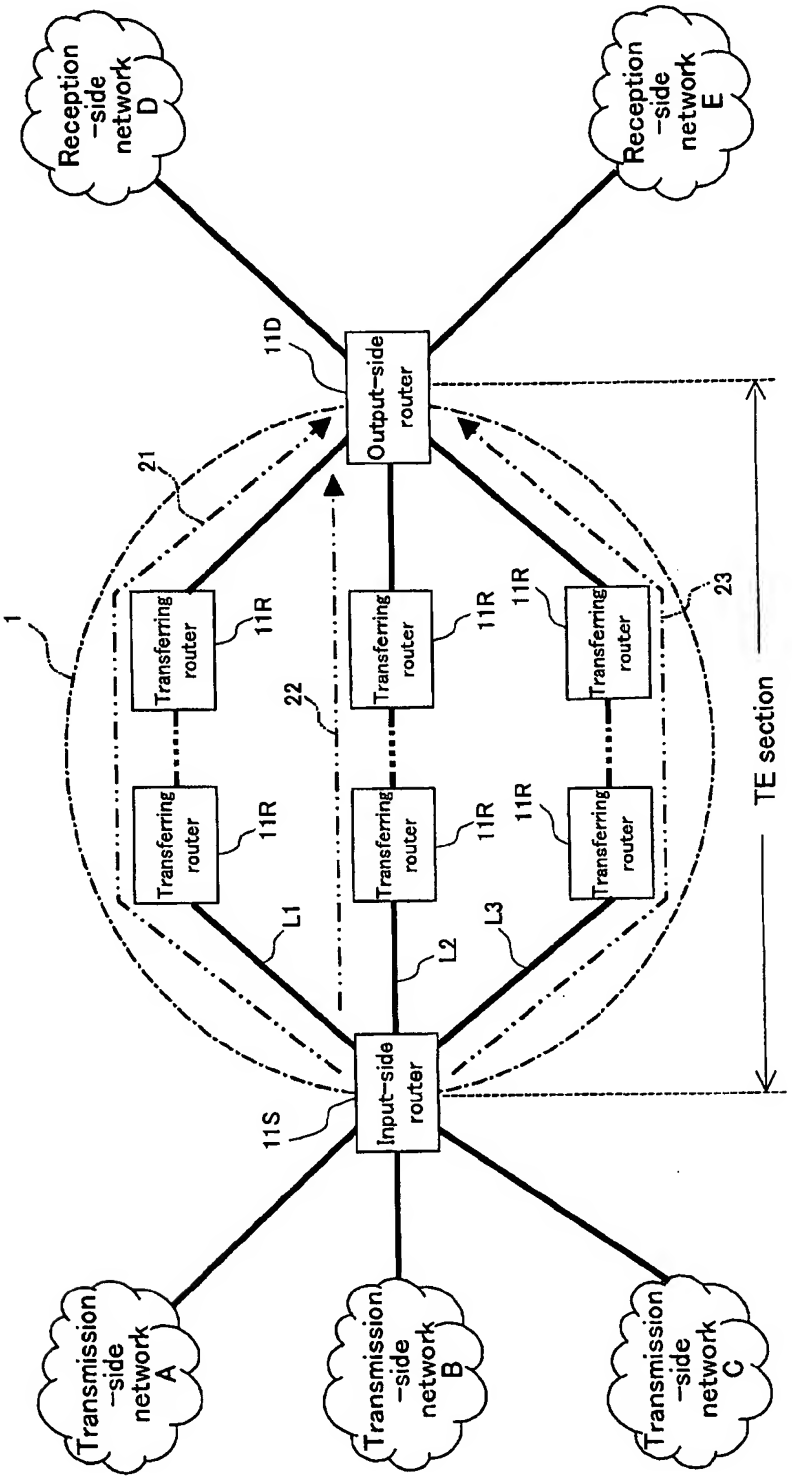

Reception -side network D
Reception -side network E
11D
Output-side router
21
1
Transferring router
11R
22
23
TE section
L1
L2
L3
Input-side router
11S
Transmission -side network A
Transmission -side network B
Transmission -side network C

FIG. 1

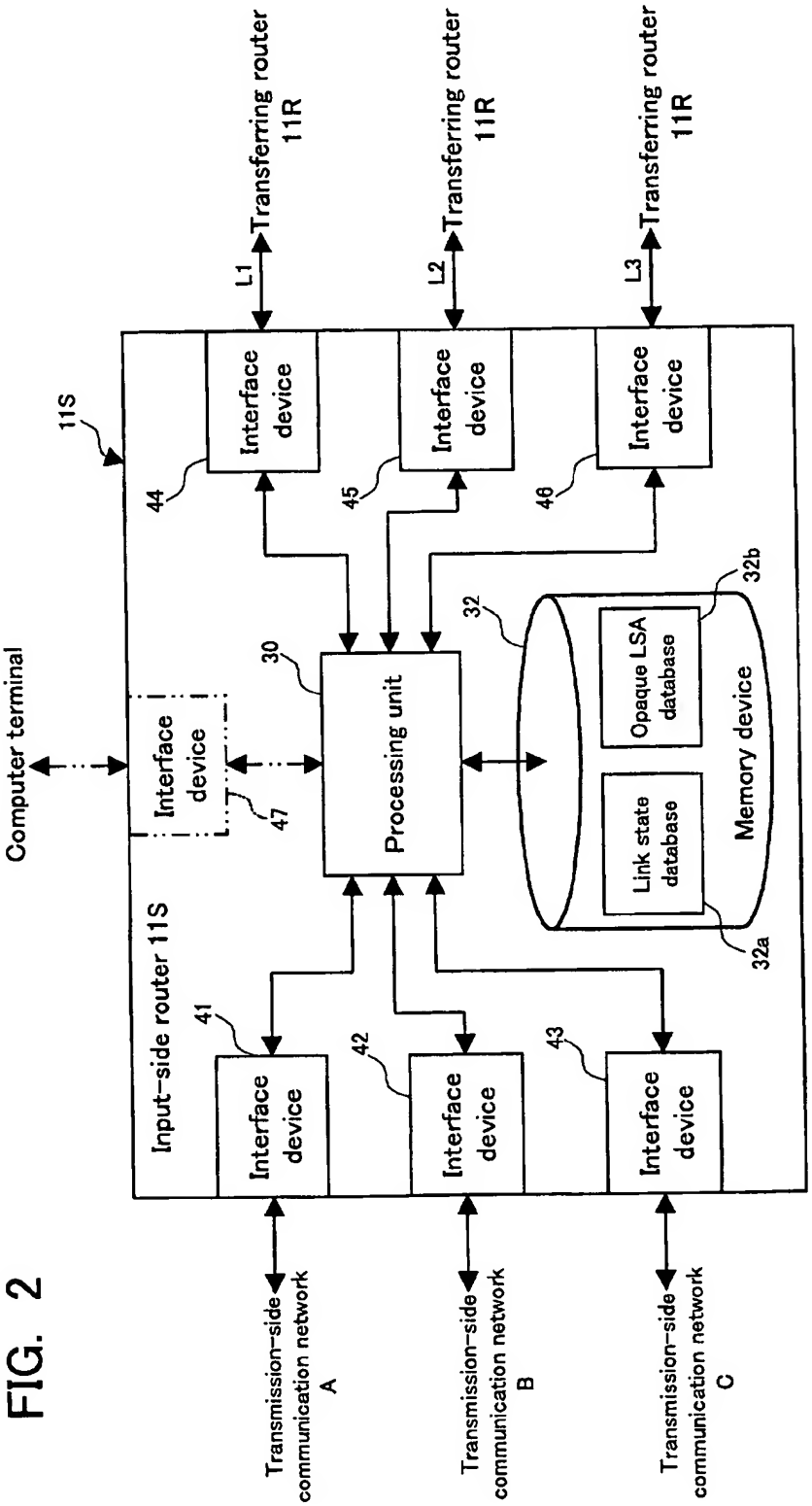

Transferring router 11R
Transferring router 11R
Transferring router 11R
L1
L2
L3
11S
Interface device
Interface device
Interface device
44
45
46
Computer terminal
Interface device
47
30
Processing unit
32
32b
Opaque LSA database
Link state database
Memory device
32a
Input-side router 11S
41
42
43
Interface device
Interface device
Interface device
Transmission-side communication network A
Transmission-side communication network B
Transmission-side communication network C

FIG. 2

FIG. 3

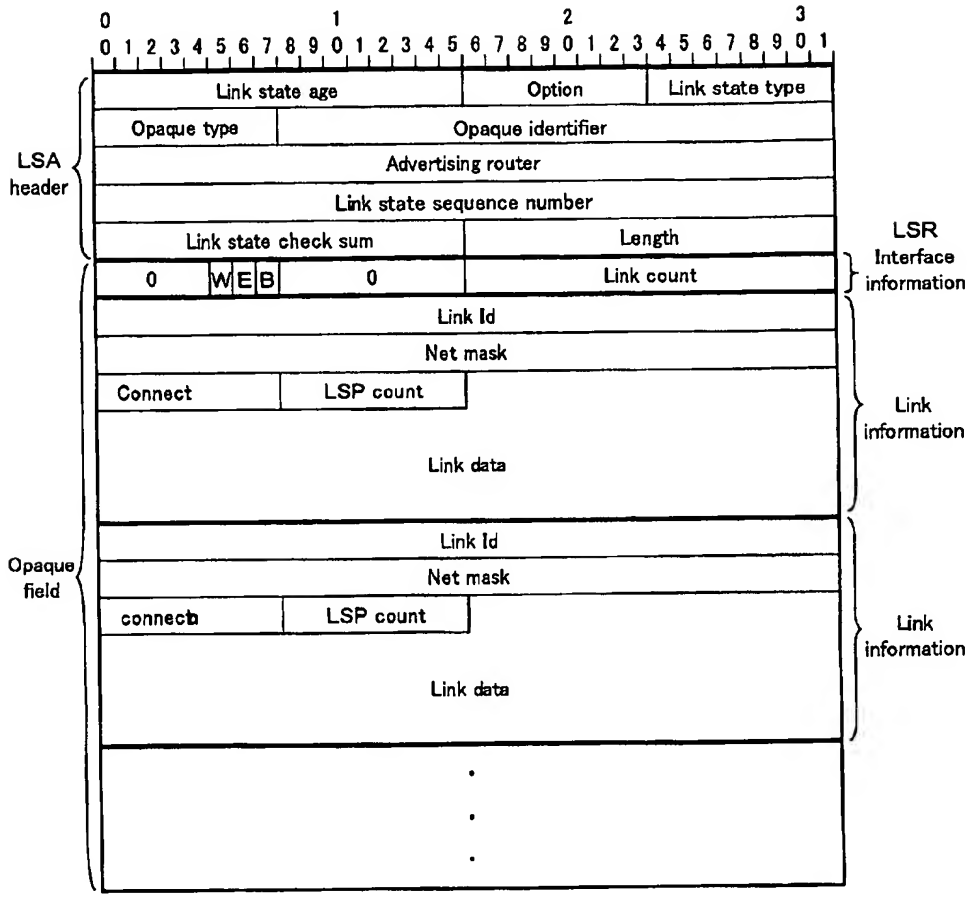

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Link state age
Option
Link state type
Opaque type
Opaque identifier
Advertising router
Link state sequence number
Link state check sum
Length
0
W
E
B
0
Link count
Link Id
Net mask
Connect
LSP count
Link data
Link Id
Net mask
connect
LSP count
Link data
LSA header
Opaque field
LSR Interface information
Link information
Link information

FIG. 4

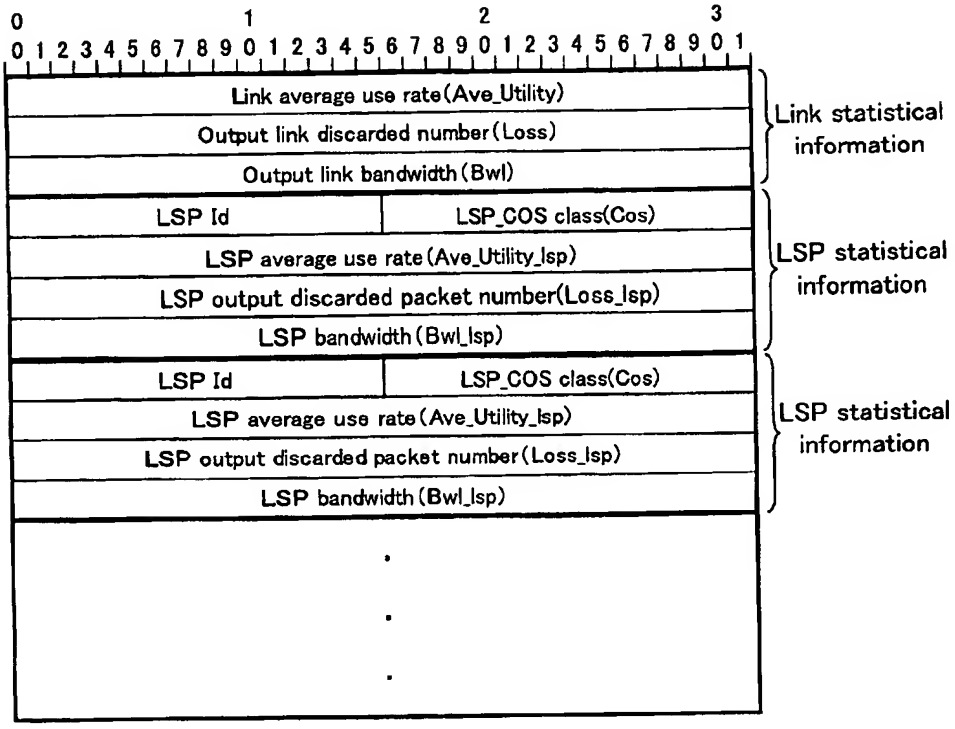

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Link average use rate(Ave_Utility)
Output link discarded number(Loss)
Output link bandwidth(Bwl)
LSP Id
LSP_COS class(Cos)
LSP average use rate(Ave_Utility_lsp)
LSP output discarded packet number(Loss_lsp)
LSP bandwidth(Bwl_lsp)
LSP Id
LSP_COS class(Cos)
LSP average use rate(Ave_Utility_lsp)
LSP output discarded packet number(Loss_lsp)
LSP bandwidth(Bwl_lsp)
Link statistical information
LSP statistical information
LSP statistical information

FIG. 5

/proc/net/dev file

interface	Receive								Transmit							
	bytes	packets	errs	drop	fifo	frame	compressed	multicast	bytes	packets	errs	drop	fifo	colls	carrier	compressed
lo	6084	67	0	0	0	0	0	0	6084	67	0	0	0	0	0	0
eth0	19954	324	0	0	0	0	0	0	26630	267	0	0	0	0	0	0
eth1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
eth2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
atm0	10488	76	0	0	0	0	0	0	5054	83	0	0	0	0	0	0
atm1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
atm2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FIG. 6

/proc/atm/device file

interface type	ESI/"MAC" addr	AAL (Tx, err, Rx, err, drop)	
0 eni	0020ea00co84	0(0 0 0 0 0)	0(83 0 76 0 0)
1 eni	0020ea005ee2	0(0 0 0 0 0)	0(0 0 0 0 0)
2 eni	0020ea005e34	0(0 0 0 0 0)	0(0 0 0 0 0)

FIG. 7

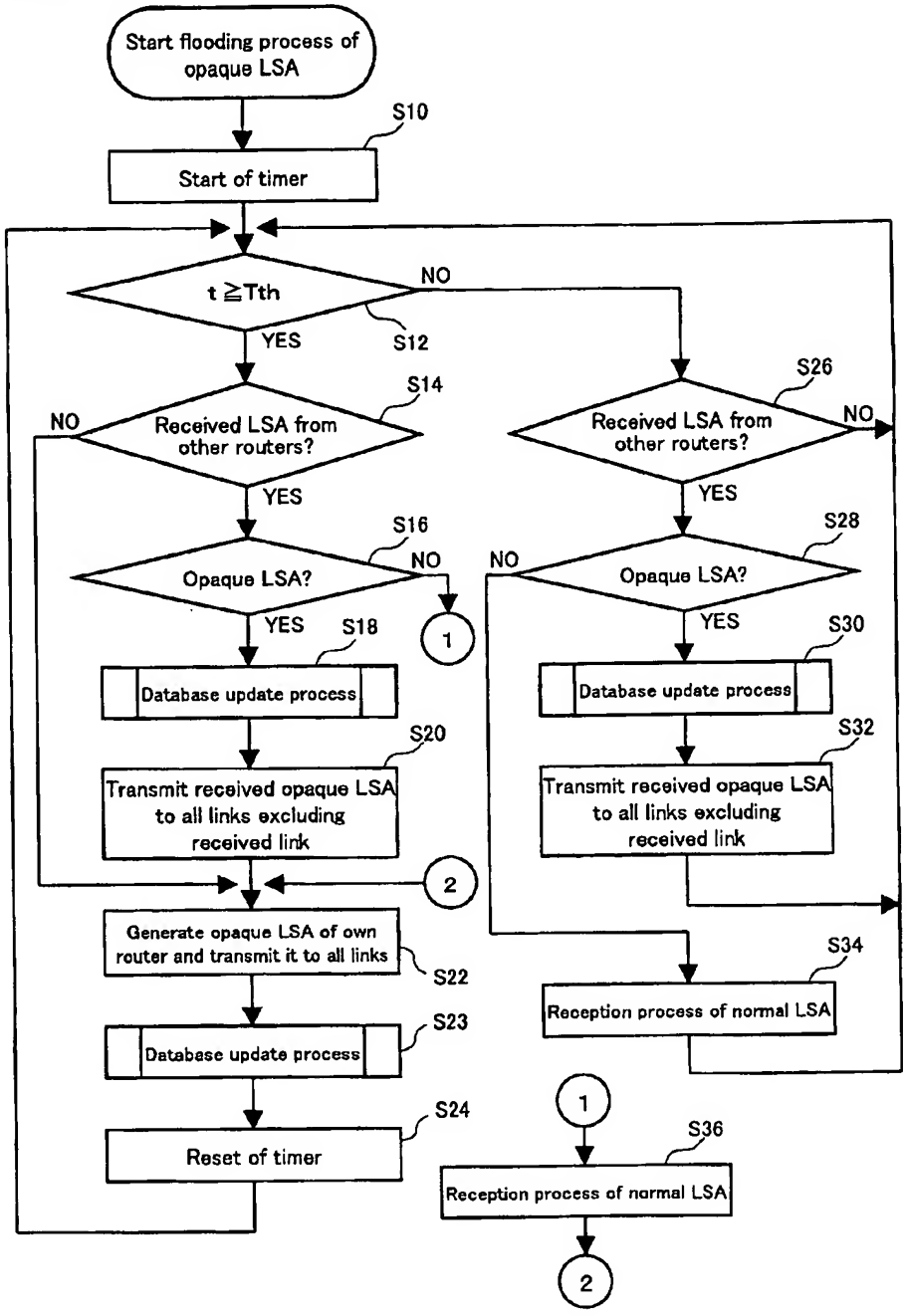

Start flooding process of opaque LSA
S10
Start of timer
t ≧Tth
NO
YES
S12
S14
Received LSA from other routers?
NO
YES
S16
Opaque LSA?
NO
YES
S18
Database update process
S20
Transmit received opaque LSA to all links excluding received link
1
2
Generate opaque LSA of own router and transmit it to all links
S22
Database update process
S23
Reset of timer
S24
S26
Received LSA from other routers?
NO
YES
S28
Opaque LSA?
NO
YES
S30
Database update process
S32
Transmit received opaque LSA to all links excluding received link
S34
Reception process of normal LSA
1
S36
Reception process of normal LSA
2

FIG. 8

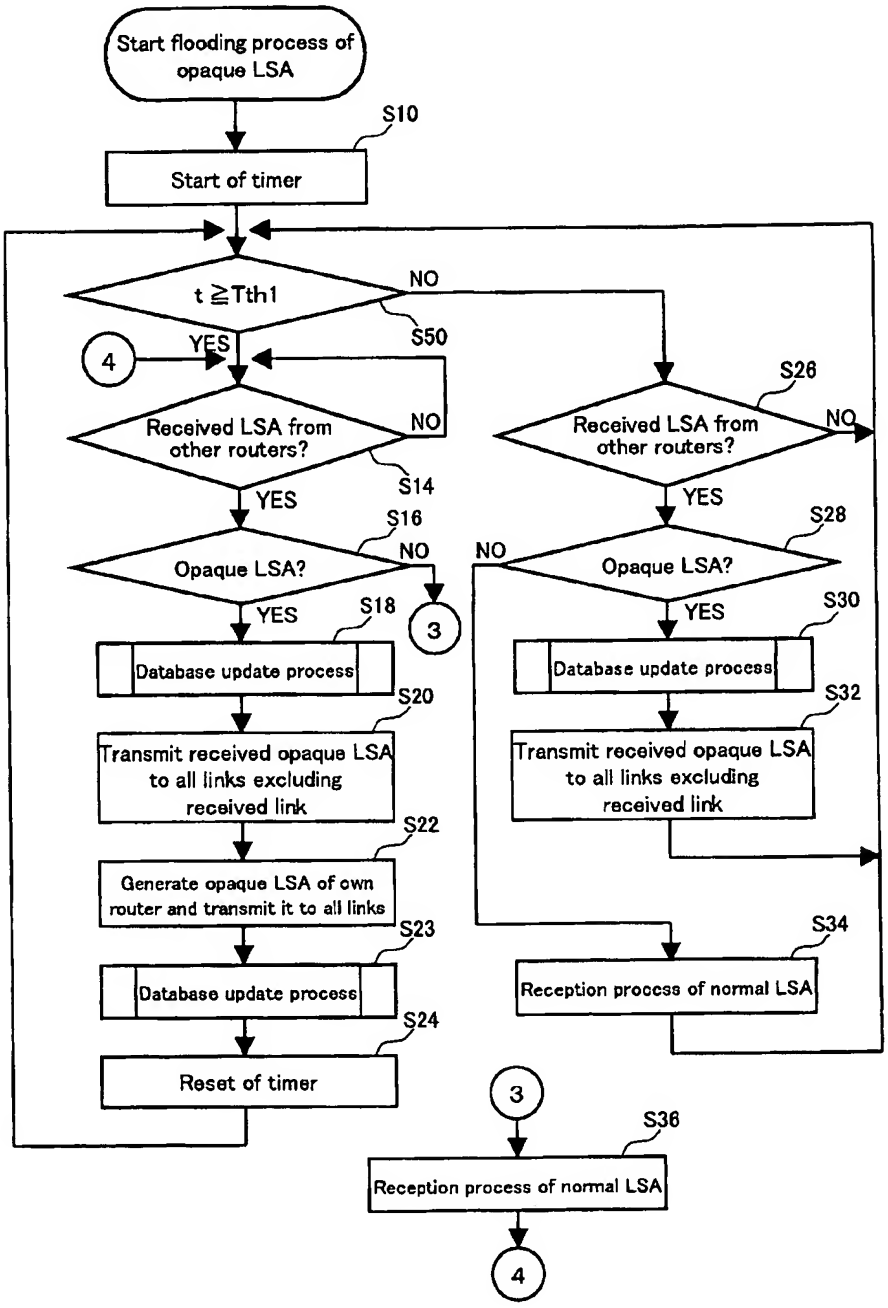

Start flooding process of opaque LSA
S10
Start of timer
t ≧ Tth1
NO
S50
YES
4
Received LSA from other routers?
NO
S14
YES
S16
Opaque LSA?
NO
YES
S18
Database update process
S20
Transmit received opaque LSA to all links excluding received link
S22
Generate opaque LSA of own router and transmit it to all links
S23
Database update process
S24
Reset of timer
3
S26
Received LSA from other routers?
NO
YES
S28
NO
Opaque LSA?
YES
S30
Database update process
S32
Transmit received opaque LSA to all links excluding received link
S34
Reception process of normal LSA
3
S36
Reception process of normal LSA
4

FIG. 9

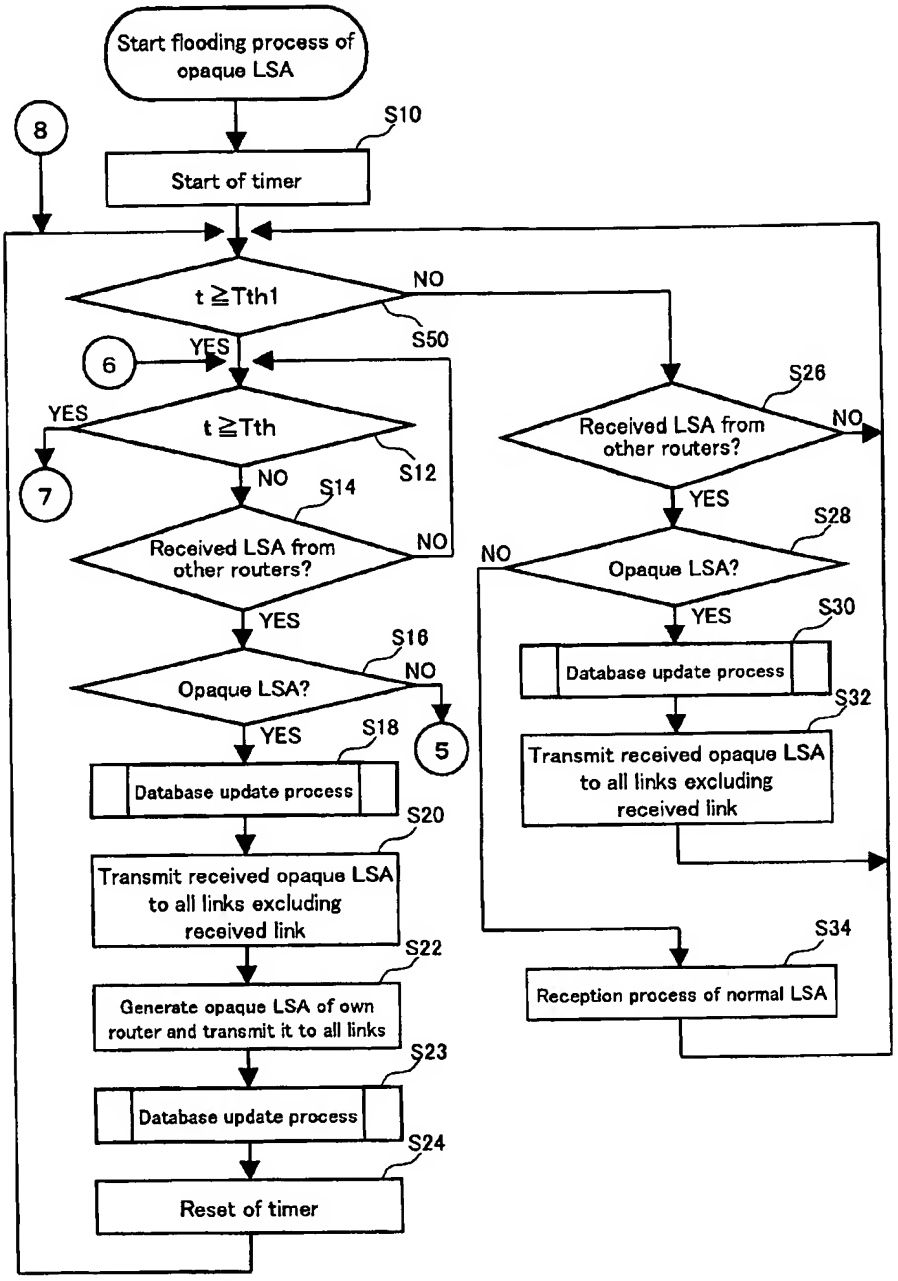

Start flooding process of opaque LSA
8
S10
Start of timer
t ≧ Tth1
NO
S50
YES
6
YES
t ≧ Tth
S12
NO
7
S14
Received LSA from other routers?
NO
YES
S16
Opaque LSA?
NO
YES
S18
5
Database update process
S20
Transmit received opaque LSA to all links excluding received link
S22
Generate opaque LSA of own router and transmit it to all links
S23
Database update process
S24
Reset of timer
S26
Received LSA from other routers?
NO
YES
S28
NO
Opaque LSA?
YES
S30
Database update process
S32
Transmit received opaque LSA to all links excluding received link
S34
Reception process of normal LSA

FIG. 10

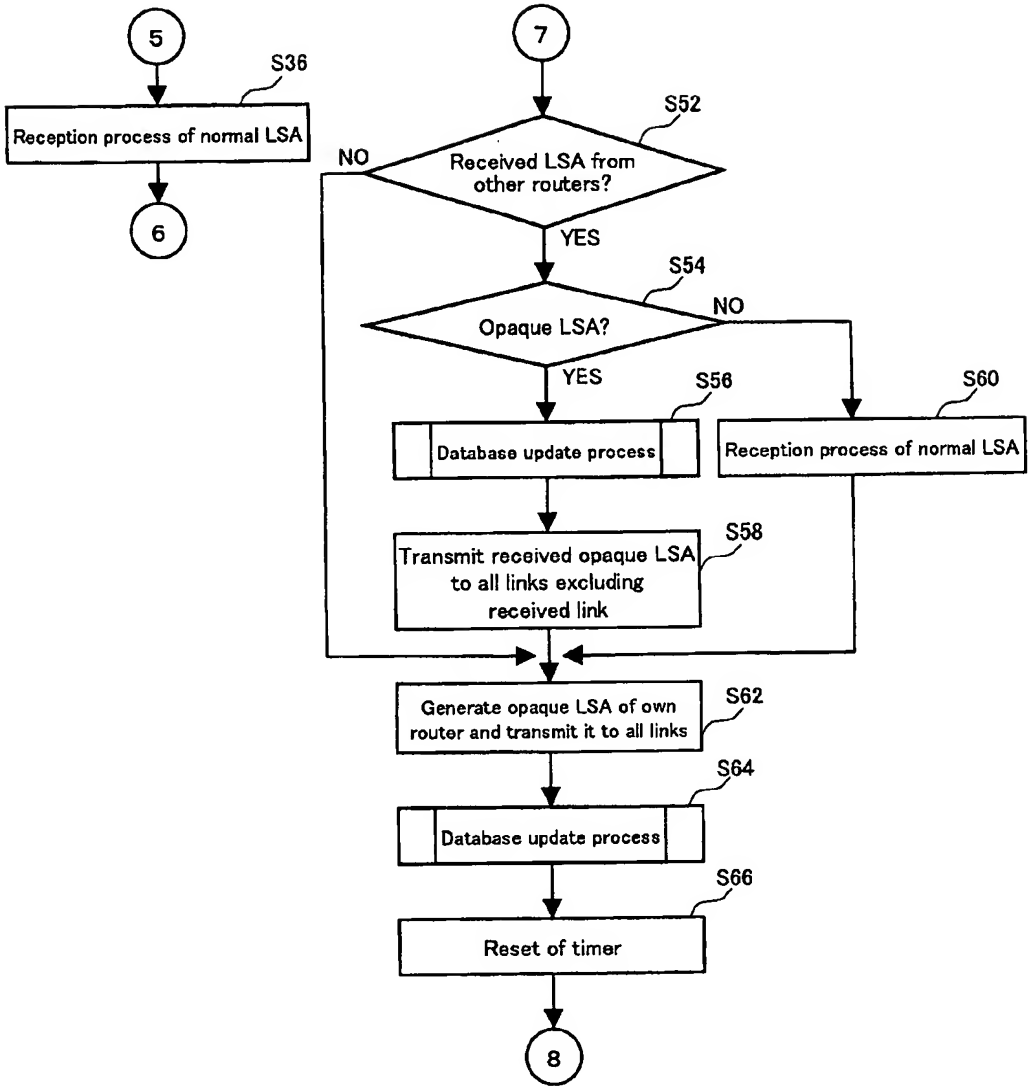

5
S36
Reception process of normal LSA
6
7
S52
Received LSA from other routers?
NO
YES
S54
Opaque LSA?
NO
YES
S56
Database update process
S60
Reception process of normal LSA
S58
Transmit received opaque LSA to all links excluding received link
S62
Generate opaque LSA of own router and transmit it to all links
S64
Database update process
S66
Reset of timer
8

FIG. 11

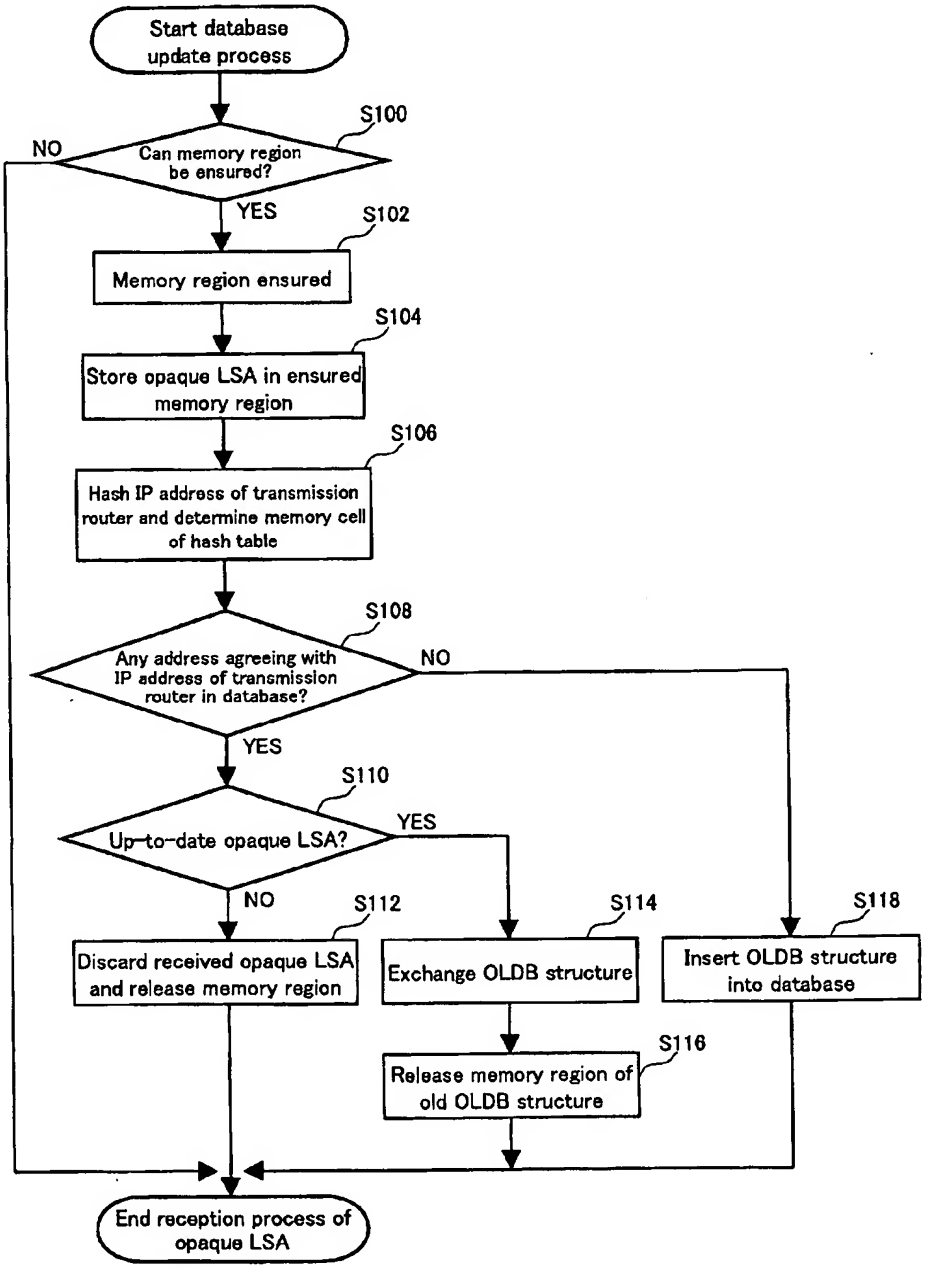
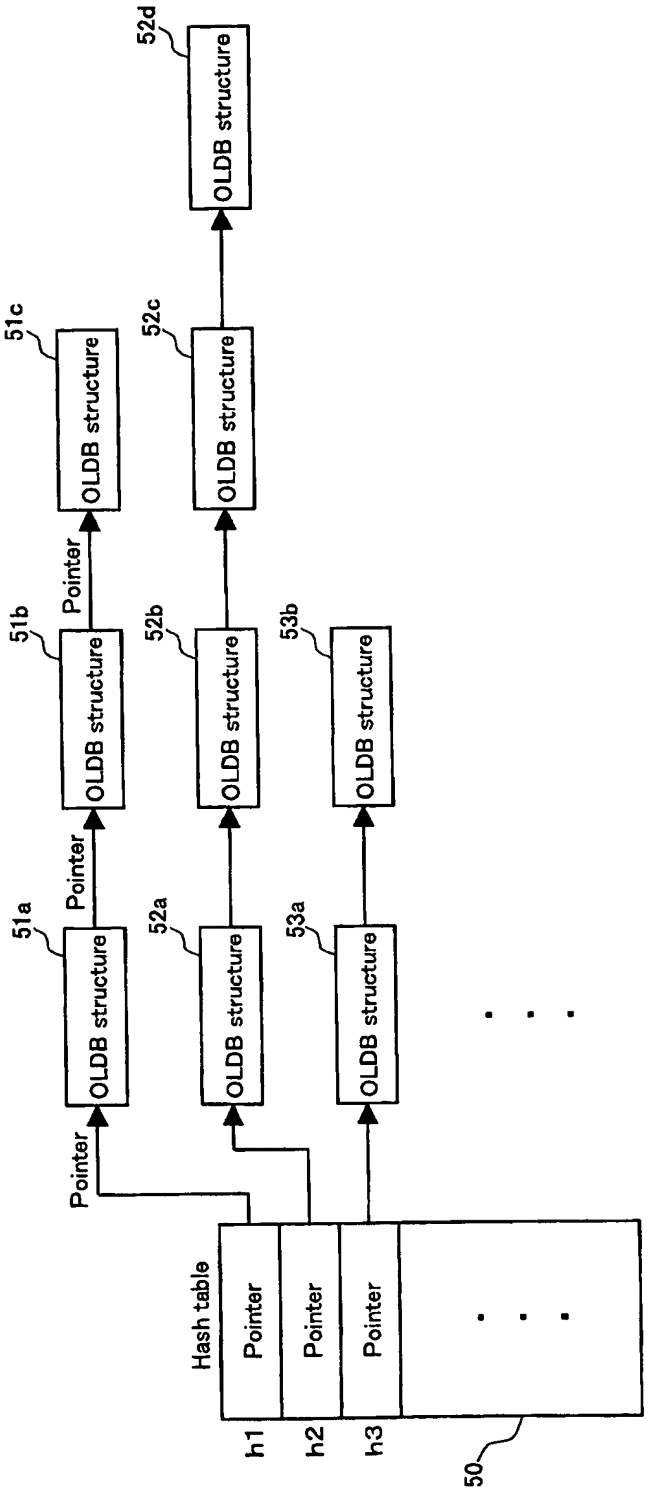

Start database update process
S100
Can memory region be ensured?
NO
YES
S102
Memory region ensured
S104
Store opaque LSA in ensured memory region
S106
Hash IP address of transmission router and determine memory cell of hash table
S108
Any address agreeing with IP address of transmission router in database?
NO
YES
S110
Up-to-date opaque LSA?
YES
NO
S112
Discard received opaque LSA and release memory region
S114
Exchange OLDB structure
S116
Release memory region of old OLDB structure
S118
Insert OLDB structure into database
End reception process of opaque LSA

FIG. 12


Hash table
Pointer
Pointer
Pointer
h1
h2
h3
50
Pointer
Pointer
Pointer
51a
51b
51c
52a
52b
52c
52d
53a
53b
OLDB structure

DEVICE AND METHOD FOR COLLECTING TRAFFIC INFORMATION

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates generally to a traffic information collecting device, a traffic information collecting method, and a traffic information collecting program product in a communication network, and more particularly, to a traffic information collecting device, a traffic information collecting method, and a traffic information collecting program product for collecting information required for making a load sharing in the communication network.

[0003] The present invention relates to a node in the communication network, having a function to collect information required for making the load sharing in the communication network.

[0004] 2. Description of the Related Art

[0005] The Internet is a connectionless network in which a connection is not established, and communication data are divided into IP packets on which an IP (Internet Protocol) address is labeled for transmission. The IP packets are transferred between routers based on the IP addresses, to deliver to a destination router or computer (hereinafter referred to simply as "an destination").

[0006] It is determined according to a routing protocol which route between the routers is used for relay based on the IP address to deliver the IP packets to the destination. As the routing protocol, at present, the RIP (Routing Information Protocol) or the OSPF (Open Shortest Path First) protocol is generally used. In these routing protocols, basically, a shortest route (or a shortest path) up to the destination is acquired based on cost of each link in the communication network (for example, a hop count), and the shortest route is set as a communication route of the IP packets.

[0007] In these routing protocols, however, the shortest route is only acquired based on cost, and a route is not acquired taking into consideration conditions of a traffic of each link. Moreover, only when a topology of the communication network changes, the new route is set, and setting or modification is not conducted in real time. On the other hand, with the spread of the Internet, a packet amount (in other words, a traffic amount or load) intercommunicating in the internet is abruptly increased.

[0008] As the result, congestion generates in the route determined by the routing protocol.

[0009] To be sure, but in the OSPF protocol there is a routing called an equal cost multipath that a plurality of the routes are determined in order to disperse the IP packets to the plurality of routes. However, this routing can be used only in the case where the plurality of routes having equal costs are present, and also as the OSPF protocol is used, it is impossible to cope with real-time fluctuations of the load of the communication network.

[0010] Then, at present, as a technology system for avoiding the congestion in the internet, traffic engineering is discussed. This traffic engineering generally sets a plurality of routes up to the destination, and also monitors conditions of the traffic of each route in real time, and selects an idle route or a route having a small load to transmit the IP packets, and disperses the load (traffic) between the plurality of routes.

[0011] In order to conduct such the traffic engineering, it is necessary that each router is informed of information in the traffic containing busy condition of each route.

SUMMARY OF THE INVENTION

[0012] With the foregoing in view, it is an object of the present invention to allow a node in a communication network to collect traffic information to thereby achieve load sharing depending on the conditions of the traffic.

[0013] In order to achieve the above object, according to a first aspect of the present invention there is provided in a communication network having a plurality of nodes, which transmit, receive or transfer communicated information, and a plurality of links (communication links), which connect the plurality of nodes to each other, a traffic information collecting device, which is provided in at least one of the plurality of nodes, for collecting traffic information all or some of the plurality of links, comprising: a traffic information collecting unit for collecting first traffic information of a first link connected to an own node among the plurality of links; a traffic information transmitting unit for transmitting, to the other nodes, said first traffic information collected by said traffic information collecting unit, using a message prescribed in a communication protocol in the communication network; a traffic information receiving unit for receiving second traffic information of second links connected to the other nodes among the plurality of links, said second traffic information being transmitted from the other nodes; and a traffic information storage for storing said first and second traffic information.

[0014] According to a second aspect of the present invention there is provided in a communication network having a plurality of nodes, which transmit, receive or transfer communicated information, and a plurality of links, which connect the plurality of nodes to each other, a traffic information collecting device, which is provided in at least one of the plurality of nodes, for collecting traffic information all or some of the plurality of links, comprising: a traffic information collecting unit for collecting first traffic information of a first link connected to an own node among the plurality of links; and a traffic information storage for storing said first traffic information collected by the traffic information collecting unit, and second traffic information of second links connected to the other nodes among the plurality of links, said second traffic information being transmitted from the other nodes.

[0015] According to the information collecting device in the present invention, the node in the communication network collects the first traffic information of the first link connected to the own node. And also the node receives the second traffic information of the second links connected to the other nodes, said second traffic information being transmitted from the other nodes. The node stores the first and second information. In consequence, the node can be informed of the conditions of the traffic of each link in the communication network based on the stored information. As the result, it becomes possible to control the load sharing in correspondence to the conditions of the traffic.

[0016] Preferably, the traffic information collecting device further comprises a traffic information transferring unit for transferring said second information to the other first links excluding the received first link.

[0017] Thus, it is possible to propagate the second traffic information transmitted to a certain node, to nodes other than the certain node as well, and the other nodes receiving the second traffic information also can obtain the traffic information. As the result, all nodes in the communication network can grasp the conditions of the traffic of each link in the communication network.

BRIEF DESCRIPTION OF THE DRAWING

[0018] **FIG. 1** is a block diagram showing a schematic configuration of a communication network according to an embodiment of the present invention;

[0019] **FIG. 2** is a block diagram showing a configuration of an input-side router;

[0020] **FIG. 3** shows a data configuration of an opaque LSA in the OSPF protocol;

[0021] **FIG. 4** shows a data configuration of a link data.

[0022] **FIGS. 5 and 6** show exemplary of data in a system status that OS of the router **11S** collects/controls in real time;

[0023] **FIG. 7** is a flowchart showing a flow of a process of the first flooding method;

[0024] **FIG. 8** is a flowchart showing a flow of a process of the second flooding method;

[0025] **FIGS. 9 and 10** are flowcharts showing a flow of a process of a third flooding method;

[0026] **FIG. 11** is a flowchart showing a flow of a detailed process of an update process of the opaque LSA database; and

[0027] **FIG. 12** shows a data configuration of an opaque LSA database.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0028] Configuration of Communication Network **1**

[0029] **FIG. 1** is a block diagram showing a schematic configuration of a communication network **1** having a node including "an information collecting device" according to the present invention. The communication network **1** forms a part of the Internet in this embodiment, and also is a communication network which performs a load sharing by traffic engineering (hereinafter called "TE"). Furthermore, the communication network **1** is, in this embodiment, a communication network that transmits and receives IP packets (hereinafter merely called "packets") in the asynchronous transfer mode (hereinafter called "ATM").

[0030] The communication network **1** is connected to transmission-side communication networks A, B, C for transmitting the packets to the communication network **1**; and reception-side communication networks D, E for receiving the packets transmitted from the communication network **1**, as external communication networks.

[0031] These external communication networks are assorted into "the transmission-side communication network" and "the reception-side communication network," for conveniences of explanation of the TE. Accordingly, the transmission-side communication network does not mean a network which only has a transmission function of the packets, and further the reception-side communication network does not mean a network which only has a reception function of the packets. For example, there is a case where the packets are transmitted from the communication network **1** to the transmission-side communication networks A, B, C, and also there is a case where the packets are transmitted from the reception-side communication networks D, E to the communication network **1**.

[0032] These transmission-side communication networks A, B, C and reception-side communication networks D, E form a part of the Internet, and in this embodiment, they are constructed by Ethernet.

[0033] The communication network **1** includes a plurality of routers (a label switch router, to be mentioned later) as one example of "a node." These routers contain an input-side router **11S**, an output-side router **11D** and a plurality of transferring routers **11R**.

[0034] When the input-side router **11S** receives the packets from the transmission-side communication network A, B, or C (or a computer terminal (not shown) connected to the input-side router **11S**), the TE is executed with respect to the packets, and the packets are transmitted to the neighboring transferring router **11R** in a route (path) set based on the TE. The transferring routers **11R** are stationed between the input-side router **11S** and the output-side router **11D**, and transfer the packets to the output-side router **11D** relaying the packets from the input-side router **11S**. The output-side router **11D** transmits the packets transmitted from the input-side router **11S** via the transferring routers **11R**, to the reception-side communication network D or E (or the computer terminal (not shown) connected to the output-side router **11D**).

[0035] Hereupon also, the routers are assorted into the input-side router **11S**, the output-side router **11D** and the transferring routers **11R** for conveniences of explanation of the TE, and for example, there is a case where the input-side router **11S** is an output-side router, or there is a case where the output-side router **11D** is an input-side router. Furthermore, hereinafter, when these input-side router **11S**, output-side router **11D** and transferring routers **11R** are not distinguished from each other, they are named generically "a router **11**" merely.

[0036] A link is provided between the routers **11** as "a communication link" (or "a transmission link") in which respective routers are physically connected to each other. For example, three links L1 to L3 are connected between the input-side router **11S** and one of neighboring three transferring routers **11R**, respectively.

[0037] In this manner, the input-side router **11S** is a starting point for executing the TE to packets transmitted from the external transmission-side communication networks A, B, C (and a computer terminal connected to the input-side router **11S**), and the output-side router **11D** is an ending point of the TE. Accordingly, a section from the input-side router **11S** to the output-side router **11D** is sometimes called "a TE section." According to this embodiment, the communication network **1** is configured so as to set a route by using MPLS (Multi-Protocol Label Switching) as

one example of a label switch technology. For this reason, the input-side router **11S**, output-side router **11D** and transferring routers **11R** are configured as a label switch router (hereinafter referred to as "LSR").

[0038] The MPLS is a technology for fusing a routing processing of an internet protocol of the layer **3** in the OSI reference model and a switching processing of the layer **2** such as ATM, frame relay, or the like. The LSR uses "a label" which is information of a lower level layer than an IP address, as packet transfer (packet switching) information. When an ATM switch is used for a packet transfer engine (label switch engine) of the LSR, VPI/VCI (Virtual Path Identifier/Virtual Connection Identifier) is used as the label. In other words, the IP addresses are mapped onto the labels (VPIs/VCIs) and a packet transfer is carried out by use of the labels.

[0039] The MPLS transfers the packets on the preset route (virtual connection) which is called the LSP (Label Switched Path). This LSP may be set along the route defined by a present routing protocol like the OSPF protocol, and may be set independently of a route by the present routing protocol. Furthermore, a plurality of the LSPs can be set up to an destination (router or computer terminal).

[0040] Accordingly, the plurality of LSPs are set independently of the present routing protocol, and a load (traffic) is dispersed between these LSPs, thereby enabling the TE.

[0041] Furthermore, in the MPLS the plurality of LSPs needs not to be preset necessarily. It is possible that a router always monitors a traffic amount in the communication network **1**, and when the traffic amount of a certain route is increased, the router searches another route directing to the same destination, sets a new LSP on the route, and shares the traffic to the new LSP. Thereby as well, it is possible to perform the dynamic TE in correspondence to a state of the traffic of the communication network.

[0042] A plurality of the LSPs can be set within one link. For example, within one link having a bandwidth of 100 [Mbps (M bit/sec)], the five LSPs with 20 [Mbps] can be set, respectively.

[0043] The setting of the LSP can be performed by reserving the bandwidth of the link in the route using, for example, RSVP (Resource Reservation Protocol) or RSVP-tunnel obtained by extending RSVP.

[0044] In the communication network **1**, as the route (the label switching path, the virtual connection in the ATM) by this LSP, as shown in **FIG. 1**, three routes **21** to **23** have been in advance set, or dynamically generated. In other words, the input-side router **11S** has the three routes **21** to **23** for transmitting the packets.

[0045] The respective three routes **21** to **23** are configured by the LSP(s) provided in one or two links or more. Furthermore, all the links configuring the respective routes **21** to **23** are not always different links, and any or all of the links are occasionally configured by the different LSPs provided in the same link.

[0046] The input-side router **11S** is positioned on the boundary of the transmission-side communication networks A, B, C outside of the TE section. The router **11S** adds a label to a packet transmitted from the transmission-side communication network A, B or C, and transmits it to the transferring router **11R**.

[0047] The transferring router **11Rs** carry out label-switching for the labeled packets, and transfer the labeled packets to the output-side router **11D**. Accordingly, the transferring router **11Rs** can transfer (label-switching transfer) the packets by only label information without retrieving the routing entry of the IP addresses of the packets.

[0048] The output-side router **11D** is positioned on the boundary of the reception-side communication networks D, E outside of the TE section. The output-side router **11D** removes the label of the packets from the transferring router **11R**, and transmits these packets to the reception-side communication network D or E (or the computer terminal of the output-side router **11D**).

[0049] As the communication network **1** employs the ATM as mentioned above, the label refers to here the VPI/VCI, and also the packets are transferred as ATM cells.

[0050] In order to make the load sharing of the traffic by setting the route of the LSP(S) in correspondence to a state of the traffic of the communication network **1**, each router **11** exchanges information in the traffic of each link and of each LSP in each link connected to itself. Hereinafter "information in the traffic" is merely referred to as "traffic information". In exchanging this traffic information, according to this embodiment, an opaque LSA (Link State Advertisement) that is a peculiar LSA in OSPF (Open Shortest Path First) protocol is used.

[0051] In the opaque LSA, there is in particular not any provision in information that the opaque LSA can include therein, excluding a LSA header and LSR interface information to be mentioned below. In this way, the opaque LSA has a freely using region. Furthermore, in the opaque LSA, there is not either any provision what time propagated, excluding a point that flooding is used as its propagation (transmission) technique. In this way each router can propagate the opaque LSA(s) at the desirable time. Moreover, as the opaque LSA follows the OSPF, the opaque LSA can be used in the communication networks such as the Internet, etc. Accordingly, the opaque LSA which has the freely using region containing the traffic information is propagated, whereby the traffic information can be exchanged between the routers in the present communication network such as the Internet, etc.

[0052] Configuration of the Router **11**

[0053] **FIG. 2** is a block diagram showing a configuration of the input-side router **11S** by representing the router **11**. The input-side router **11S** includes a processing unit **30**; a memory device **32**; and interface devices **41** to **46**. Furthermore, when the computer terminal is connected to the input-side router **11S**, a computer terminal interface device **47** is provided. Incidentally, as the transferring router **11R** and output-side router **11D** have the same configuration as the input-side router **11S**, excluding a case where the number of interface device(s) is (are) different and a case where the connection destination(s) is (are) different, only the configuration of the input-side router **11S** will be explained hereupon.

[0054] The memory device **32** can be configured by a hard disk device, etc., and contains a link state database **32*a*** and an opaque LSA database **32*b***. The link state database **32*a*** is a database defined by the OSPF protocol and stores link

states. The opaque LSA database **32***b* is a database storing the opaque LSAs, and the detailed description will be mentioned later.

[0055] The processing unit **30** has a CPU, a memory (RAM, ROM, etc.), and the like (not shown). This processing unit **30** controls the memory device **32** and the interface devices **41** to **46** (**47**) in accordance with a program stored in the internal memory or the memory device **32**. And also the processing unit **30** carries out setting of routes, control of the load sharing in correspondence to a state of the traffic, transmission and reception of the packets (ATM cells), generation, transmission and reception of the opaque LSAs, update of the link state database by the received opaque LSAs, update of the opaque LSA database by the received opaque LSA, or the like according to the program.

[0056] The interface devices **41** to **46** (**47**) contains an I/O buffer, and process I/O of the packet (ATM cell) for transferring respective connected links under the control of the processor **30**.

[0057] A hardware switch configured by a hardware circuit is provided in the router **11S**, and can transmit and receive the packets (ATM cells) (containing generation of the ATM cells and switching of the labels (VPIs/VCIs)).

[0058] Data Configuration of the Opaque LSA

[0059] The traffic information is flooded between the routers **11** by use of the opaque LSA(s) in the OSPF protocol, as mentioned above. This opaque LSA has a data configuration shown in **FIG. 3**. In **FIG. 3**, a digit designated in an uppermost part is a bit number. Accordingly, **FIG. 3** shows that the opaque LSA stands in a row in 32 bits (4 bytes) unit.

[0060] There is not any provision of the opaque LSA, excluding the fields of the LSA header and LSR interface information, and the opaque LSA has a freely using field (a field of link information of **FIG. 3**). Accordingly, according to this embodiment, the traffic information is flooded between the routers **11** by use of the opaque LSAs having the data configuration shown in **FIG. 3**.

[0061] The opaque LSA is assorted largely into the LSA header and an opaque field.

[0062] The LSA header has 20 bytes, and has the same data configuration as the normal LSA (i.e., the LSA other than the opaque LSA). In other words, the LSA header is constructed by respective fields such as a link state age, an option, a link state type, an opaque type, an opaque identifier (an opaque ID), an advertising router, a link state sequence number (LS sequence number), a link state checksum (LS checksum), and a length.

[0063] "The link state age" has 2 bytes, and designates a lifetime (second unit) of this opaque LSA, namely a valid period.

[0064] "The option" has 1 byte, and is a field where it is set so that the router supports an optional function, or the support level can be transmitted to the other routers. A second bit of this "option" field is called 0 bit, and designates whether or not the router supports the opaque LSA (is opaque-capable).

[0065] "The link state type" has 1 byte, and designates a type of LSA. In the opaque LSA, a value of the link state type is 9, 10 or 11, and a flooding scope differs corresponding to this value.

[0066] When this value is 9, the flooding scope of the opaque LSA is "link-local," and within a local (sub)network. In the case of 10, the flooding scope is "area-local," and the opaque LSA is not flooded beyond the area that it is originated into. In the case of 11, the flooding scope is "equivalent to AS-external LSA," and the opaque LSA is flooded throughout the AS (Autonomous System). In particular, (1) the opaque LSA is flooded in all transit area; (2) the opaque LSA is not flooded from backbones to stub areas; and (3) the opaque LSA is not generated from a certain router into stub areas connected to the router.

[0067] "The advertising router" has 4 bytes, and is a field storing an IP address of the router that creates and advertises the opaque LSA. The link state number and link state checksum are used in the case where 2 or more opaque LSAs created by the same advertising router are present, when it is determined which one is up-to-date (latest temporally). "The length" is has 2 bytes, and designates a length (a byte number) of the opaque LSA.

[0068] "The opaque field" includes LSR interface information and one or more sets of link information. The number of the sets of the link information is equal to the number of links connected to the router **11**.

[0069] The LSR interface information has 4 bytes, and is configured by an upper 2-byte E_B field and a lower 2-byte link count field. "The E_B field" designates whether the router is an area border router or an AS boundary router. "The link count" designates the number of links connected to the router.

[0070] "The link information" is configured by respective fields such as a link identifier, a net mask, a connection type, a LSP count, and a link data.

[0071] "The link identifier" has 4 bytes, and designates the IP address of the neighboring router or computer terminal connected to the link. "The net mask" has 4 bytes, and designates a (sub-)net mask of the link identifier. "The connection type" has 1 byte, and designates whether a destination of the link is a router, a computer terminal, or the like. The LSP count has 1 byte, and designates the number of LSP provided in the link. This value dynamically changes as the number of LSP during communication changes.

[0072] "The link data" is configured by a plurality of data. **FIG. 4** shows a data configuration of the link data. The link data have link statistical information of the link and LSP statistical information of each LSP formed in the link. These link statistical information and LSP statistical information are one example of "information in the traffic (traffic information)" pertaining to the present invention.

[0073] "The link statistical information" has an link average usage rate field (Ave_Utilization), an output link discarded packet number field (Loss), and an output link bandwidth field (Bwl).

[0074] "The link average usage rate" has 4 bytes, and designates an average usage rate of the link. This link average usage rate Ave_Utilization is calculated by the following equation (1):

$$Ave\_Utilization=\alpha \times CUTY(n)+(1-\alpha)\times CUTY(n-1) \quad (1)$$

[0075] This equation (1) is a calculation equation by a moving average. Here, a is a smoothing coefficient. CUTY(n) designates a present (namely, time n) link usage rate, and CUTY(n−1) designates a link usage rate of one time past (namely, time (n−1)), respectively, and they are obtained by the following equation (2):

$$CUTY(n)=[\text{the number of packets outputted to the output link from time } (n-1) \text{ to time } n]/[\text{Bandwidth of the output link}] \quad (2)$$

[0076] The average usage rate may be set as a maximum of the observed CUTY, and in this case, the average usage rate is represented by the following equation (3):

$$Ave\_Utilization=\max(\text{observed value}) \quad (3)$$

[0077] "The output link discarded packet number" has 4 bytes, and designates the total number of the discarded output packet of the link. The total number is calculated by the following equation (4):

$$Loss(n)=Loss(n-1)+NLoss \quad (4)$$

[0078] Here, Loss(n−1) is the total number of the output link discarded packets up to time (n−1) of one time past, and NLoss is the number of output link discarded packets generated from time (n−1) to time n.

[0079] "The output link bandwidth" has 4 bytes, and designates bandwidth [bps (bit/sec)] of an output link. This output link bandwidth has been in advance stored in a memory of the processing unit **30** or the memory device **32** of each router **11**, and the stored value is written in the field of the opaque LSA.

[0080] "The LSP statistical information" has a LSP identifier field (Lsp_id), a LSP_COS class field (Cos_lsp), a LSP average usage rate field (Ave_Utility_lsp), a LSP output discarded packet number field (Loss_lsp), and a LSP bandwidth field (Bwl_lsp).

[0081] "The LSP identifier" has 2 bytes, and designates an identifier of the LSP. "The LSP_COS class" has 2 bytes, and designates the COS class of the LSP.

[0082] "The LSP average usage rate" has 4 bytes, and designates an average usage rate of the LSP, and is obtained by the equation that the output link is replaced by not a link, but LSP in the equations (1) and (2). Furthermore, similarly to the equation (3), it can be set to a maximum of the observed value in the LSP.

[0083] "The LSP output discarded packet number" has 4 bytes, and designates the total number of the discarded output packet of the LSP, and is obtained by replacing the output link by the LSP in the equation (4).

[0084] "The LSP bandwidth" designates bandwidth [bps] allocated when the LSP is set. This LSP bandwidht is stored in the memory of the processing unit **30** or the memory device **32** of the router **11** that has output links in which the LSPs are set, at the time of setting the LSP. The LSP bandwidth value in the memory is written in the field of the opaque LSA.

[0085] The above-mentioned link average usage rate, output link discarded packet number, LSP average usage rate, and LSP output discarded packet number are obtained on the basis of data (file) representing a system status (CPU load, the number of transferring packets, etc.). These data are, for example, collected and managed in real time by an operating system (OS: for example, Linux) of each router **11**.

[0086] **FIGS. 5 and 6** show examples of data in a system status that OS of the router **11S** collects/manages in real time. **FIG. 5** shows a content of the file in a directory "/proc/net/dev" of the memory device **32** provided in the router **11S** in the form of table. **FIG. 6** shows a content of the file in a directory "/proc/atm/device" of the memory device **32** in the form of table.

[0087] Figures shown in these drawings are examples of data at a certain time. Furthermore, these figures designate integrated values from when the router **11S** is turned on, and as OS collects in real time (fixed time interval Δt), they change every moment. As these figures are integrated values, for example, by acquiring a difference between a value when referring to this table at time (n−1) and a value when referring to this table at time n, it is possible to acquire variation between 1 time from time (n−1) to time n.

[0088] The upper half part of the table shown in **FIG. 5** designates data in reception of the router **11S** as shown by a character of "Receive", and the lower half part designates data in transmission of the router **11S** as shown by a character of "Transmit." Hereinafter, the upper half part of the table is called "a reception table," and the lower half part is called "a transmission table."

[0089] The "interface" in the reception table and transmission table designates the kind of interface connected to the router **11S**. As this interface, thereare "lo,""eth0,""eth1," "eth2,""atm0,""atm1," and "atm2."

[0090] "lo" is an abbreviation of "loopback device," and means an interface with the computer terminal (not shown in **FIG. 1**) connected to the router **11S**. "eth0" to "eth2" mean the interface with the Ethernet. As the transmission-side communication networks (Ethernet) A to C are connected to the router **11S**, as shown in **FIG. 1**, "eth0" means the interface with the transmission-side communication network A, and "eth1" means the interface with the transmission-side communication network B, and "eth2" means the interface with the transmission-side communication network C, respectively.

[0091] "atm0" to "atm2" mean the interface with the ATM. As shown in **FIG. 1**, as three links L1 to L3 of the communication network **1** configured by the ATM network are connected to the input-side router **11S**, "atm0" means the interface with the link L1, "atm1" means the interface with the link L2, and "atm2" means the interface with the link L3, respectively.

[0092] The "bytes" of the reception table and transmission table designates a byte number received from each interface and a byte number transmitted to each interface, respectively. For example, it is indicated from the data of the interface "lo" that the input-side router **11S** receives data of 6084 bytes from the computer terminal, and transmits the data of the same byte number to the interface "lo."

[0093] The "packets" of the reception table and transmission table designates a packet number received from each interface and a packet number transmitted to each interface, respectively. For example, it is indicated from the data of the interface "eth0" that the router **11S** receives 324 packets from the transmission side communication network A and transmits 267 packets thereto.

[0094] The "errs" of the reception table and transmission table designates an error number of the reception packet from each interface and an error number of the transmission packet to each interface, respectively.

[0095] The "drop" of the reception table and transmission table designates a discard number of the reception packet from each interface and a discard number of the transmission packet to each interface, respectively.

[0096] The "fifo" of the reception table and transmission table designates a packet number (length of queue (FIFO)) of a reception process waiting and a packet number (length of queue (FIFO)) of a transmission process waiting, respectively.

[0097] The "frame" of the reception table designates a reception frame number, and the "compressed" designates a compressed packet number, and the "multicast" designates a reception multicast packet number, respectively.

[0098] The "colls" of the transmission table designates a generation number of collision in the Ethernet (CSMA/CD system), and the "carrier" designates a carrier detection number in the Ethernet, and the "compressed" designates a compressed packet number, respectively.

[0099] The "interface type" of the table shown in **FIG. 6** designates three types of ATM interface of the links L1 to L3 of the communication network **1**, and "0 eni" is No. 1 of interface name "eni," and "1 eni" is No. 2 of interface name "eni," and "2 eni" is No. 3 of interface name "eni."

[0100] The "ESI/"MAC addr"" designates a MAC (Media Access Control) address. The "AAL (Tx, err, Rx, err, drop)" designates statistical information in the packet of an ATM adaptation layer. The "Tx" designates a transmission packet number, and a left side "err" designates a transmission error number, and the "Rx" designates a reception packet number, and a right side "err" designates a reception error number, and the "drop" designates a reception discard packet number, respectively.

[0101] Both **FIGS. 5 and 6** show data in the link, and the OS can collect data in each LSP provided in the link.

[0102] As this system state is collected in real time (each fixed time interval Δt) by the OS, data of each field of the above-mentioned link data are acquired based on these collected information.

[0103] For example, by acquiring a difference between "packets" of the transmission table of **FIG. 5** at time (n-1) and "packets" of the transmission table at time n, an output packet number of the equation (2) is acquired. Furthermore, the output packet number which is the difference is divided by the bandwidth of the link, whereby CUTY(n) of the equation (2) is acquired. Furthermore, Ave_Utilization of the equation (1) is acquired based on this CUTY(n).

[0104] Furthermore, as the transmission table of **FIG. 5** or the "drop" of **FIG. 6** is an integrated value of the discard packet number, a value of the "drop" at time n is Loss (n) of the equation (4).

[0105] Flooding Method of the Opaque LSA

[0106] (1) First Flooding Method

[0107] The opaque LSA is transmitted to other routers by use of the flooding. A timing of the flooding can be set, for example, in each fixed time interval Tth. This fixed time interval Tth is equal to or more than the time interval Δt when the system status is updated (preferably, the time interval Tth is equal to a positive integer times Δt), and is set to an interval that the communication network **1** is not congested by transmission of the opaque LSAs, and also an interval that the TE can effectively be conducted. The specific value is acquired by experiments, simulations, computations, etc.

[0108] **FIG. 7** is a flowchart showing a flow of a process of a first flooding method. This process is executed by the processing unit **30** (refer to **FIG. 2**).

[0109] First, when the router **11** is activated by turning on the router **11**, etc., a timer (not shown) provided inside the processing unit **30** starts timing (step **S10**).

[0110] Subsequently, it is determined whether or not time t of the timer reaches the flooding time interval Tth (step **S12**). When not reaching it (NO in step **S12**), it is determined whether or not the LSAs are received from the other routers **11** (step **S26**). When the LSAs are not received from the other routers (NO in step **S26**), returning to step **S12**, and when received (YES in step **S26**), it is determined whether or not the received LSA is an opaque LSA (step **S28**). This determination is made by the value of the link state type of the above-described LSA header. When the link state type is **9**, **10** or **11**, it is determined that it is the opaque LSA, and when other than those, it is determined that it is a normal LSA (LSA other than the opaque LSA).

[0111] In the case of the opaque LSA (YES in step **S28**), the opaque LSA database is updated (to be mentioned below) (step **S30**), and after that, the received opaque LSA is transmitted (flooded) to all the links (output links) other than the received link (step **S32**). After that, the process will return to step **S12**. On the other hand, in the case of the normal LSA (NO in step **S28**), a reception process of the normal LSA is performed (step **S34**), and after that, the process is returned to step **S12**. This reception process of the normal LSA contains an update process of the link state database.

[0112] When time t of the timer is time Tth or more in step **S12** (YES in step **S12**), it is determined whether or not the LSA is received from the other routers (step **S14**).

[0113] When the LSA is received (YES in step **S14**), the processes of steps **S16** to **S20**, **S36** are performed. These processes are similar to those in steps **S28** to **S32**, **S34**, as above, respectively. Then, as the timer time t elapses in a fixed time interval Tth, the processing unit **30** creates the opaque LSA of its own router, and the opaque LSA is transmitted (flooding) to all the links (output links) (step **S22**).

[0114] Subsequently, the update process of the opaque LSA database by the opaque LSA of the own router is performed (step **S23**). And, the timer is reset to 0. Then the process will return to step **S12**.

[0115] On the other hand, in step **S14**, when the opaque LSA is not received from the other routers, the opaque LSA of the own router is generated immediately, and is transmitted to all the links (step **S22**). Then, the update process of the opaque LSA database by the opaque LSA of the own router is performed (step **S23**). And, the timer is reset (step **S24**), and the process will return to step **S12**.

[0116] (2) Second Flooding Method

[0117] Among the flooding methods, there is a method in which when the opaque LSA is received from the other routers, the received opaque LSA is transmitted, and also the opaque LSA which accommodates traffic information of the own router is generated to transmit. This method is called "a second flooding method".

[0118] **FIG. 8** is a flowchart showing a flow of a process of the second flooding method. The same symbol is labeled to the same process as that in the above-mentioned flowchart of **FIG. 7**, and the detailed description is omitted. The processing unit **30** executes this process.

[0119] First, when the router **11** is turned on, etc. and starts, the timer starts (step **S10**), and it is determined whether or not the time t of the timer reaches threshold time Tth1 (step **S50**). This threshold time Tth1 is set to a time interval which is time interval Δt or more when the OS of the router **11** updates the system status, and which is smaller than the time interval Tth in the first flooding method. The reason why the time Tth1 is set as above is that even when two or more opaque LSAs are received from the other routers before this fixed time Tth1 elapses, the opaque LSAs of the same content in the own router are not flooded twice or more.

[0120] When the time t of the timer reaches the threshold time Tth1 (YES of step **S50**), the processing unit **30** waits for receiving the opaque LSA from the other routers **11** (step **S14**). After receiving the opaque LSA, the processing unit **30** creates the opaque LSA which accommodates the traffic information of its own router and transmits it (steps **S14** to **S24, S36**). Subsequently the process is returned to step **S50**.

[0121] On the other hand, when the time t of the timer does not reach the threshold time Tth1 (NO of step **S50**), even if the own router **11** receives the opaque LSA from the other routers **11**, the own router **11** does not create the opaque LSA which accommodates the traffic information of itself and transmit it (steps **S26** to **S34**). Subsequently the process is returned to step **S50**.

[0122] (3) Third Flooding Method

[0123] There is a third flooding method in which the first flooding method and the second flooding method are composited. In other words, the opaque LSAs are transmitted at the fixed time interval Tth (a first flooding method), and also when the own router receives the opaque LSA from the other routers **11**, it transmits the received opaque LSA, and also creates the opaque LSA which accommodates the traffic information of itself to transmit (a second flooding method).

[0124] **FIGS. 9 and 10** are flowcharts showing a flow of a process of the third flooding method. The same symbol is labeled to the same process as that in the above-mentioned flowchart of **FIGS. 7 and 8**, and the detailed description is omitted. The processing unit **30** executes this process.

[0125] After the timer starts (step **S10**), it is determined whether or not the time t of the timer elapses in the fixed time Tth1 as mentioned in the second flooding method (step **S50**).

[0126] When the time t of the timer does not reach the fixed time Tth1 (NO of step **S50**), the processes in steps **S26** to **S34** are executed. In other words, when the own router receives the opaque LSA from the other routers **11**, the own router executes the database update process by the opaque LSA, and also transmits this opaque LSA to the other router. Subsequently, the process is returned to step **S50**.

[0127] On the other hand, when the time t of the timer reaches the fixed time interval Tth1 (YES of step **S50**), the timer time t is compared with the fixed time interval Tth mentioned in the first flooding method (step **S12**).

[0128] When the time t of the timer does not reach the fixed time interval Tth (NO of step **S12**), the own router waits for reception of the opaque LSA from the other routers **11**, and executes the LSA received from the other routers **11** (database update process, transmission process to the other router, or the like) (steps **S16** to **S24, S36**). Subsequently, the process is returned to step **S50**.

[0129] On the other hand, when the time t of the timer reaches the fixed time interval Tth (YES of step **S12**), the own router creates the opaque LSA which stores the traffic status of the own router irrespective of presence or absence of reception of the opaque LSA from the other routers and transmits it (steps **S52** to **S64**). Incidentally, the processes of steps **S52** to **S60** are similar to the processes of steps **S26** to **S34**, respectively, and the processes of steps **S62** to **S64** are similar to the processes of steps **S22** to **S24**, respectively. Subsequently, the process is returned to step **S50**.

[0130] Data configuration of the opaque LSA database and its update process **FIG. 12** shows a data configuration of an opaque LSA database **32***b* provided in the memory device **32** of the router **11**.

[0131] The opaque LSA database **32***b* has a hash table **50**, opaque LSA database structures (hereinafter referred to as "an OLDB structure") **51***a* to **51***c*, **52***a* to **52***d*, **53***a*, **53***b*, etc having the opaque LSA of each router (containing the own router) **11**.

[0132] The hash table **50** has a plurality of memory cells. Each memory cell is accessed with a hash value h1, h2, h3, or the like as its address. The hash value is determined by hashing the value of "an advertising router" (namely, the IP address of the advertising router) in the LSA header of the received opaque. A pointer to the OLDB structure is stored in each memory cell. For example, the pointer to the OLDB structure **51***a* is stored in the memory cell in correspondence to the hash value hl. The pointer to the OLDB structure **52***a* is stored in the memory cell in correspondence to the hash value h2, and the pointer to the OLDB structure **53***a* is stored in the memory cell in correspondence to the hash value h3, respectively. The pointers to the other OLDB structures (not shown) are also stored in the memory cells in correspondence to the other hash values (not shown).

[0133] The OLDB structure contains a content (each field shown in **FIGS. 3 and 4**) of the received opaque LSA. It also contains the pointer to the neighboring next OLDB structure (for example, the OLDB structure **51***a* contains the pointer to the next OLDB structure **51***b*), a field required for calculating a distance (cost) up to a root, etc. One router **11** corresponds to one OLDB structure.

[0134] As the hash value of the hash table **50**, for example, it is possible to use a quotient (integer value) obtained by dividing an "advertising router" value (IP address) of the LSA header deemed as an integer by a prime **251**. In this

case, the hash table **50** has **251** memory cells, and the plurality of routers **11** in the connection network **1** are classified into **251** groups. The pointers connect the OLDB structures of the plurality of routers belonging to each group classified into the **251** groups, for example, like the OLDB structures **51***a* to **51***c*.

[0135] Incidentally, as the IP addresses of the routers belonging to the same group are close, the routers are placed at geographically close positions in many cases.

[0136] In this manner, as the routers can be classified into the groups by using the hash table, as described below, when retrieving the OLDB structure of a specified router, it is possible to accelerate the retrieval.

[0137] **FIG. 11** is a flowchart showing a flow of a detailed process of an update process of the opaque LSA database in steps **S18**, **S23**, **S30** and **S64** shown in FIGS. **7** to **10**. The processing unit **30** of each router **11** executes this process.

[0138] First, the processing unit **30** determines whether or not it is possible to ensure a storage region of the OLDB structure for storing the opaque LSA within the memory device **32** (step **S100**). When it is possible to ensured (YES in step **S100**), the processing unit **30** ensures the storage region (step **S102**), and stores the received opaque LSA in the ensured storage region (step **S104**).

[0139] Subsequently, a hash value of the IP address of the router that transmitted the opaque LSA is calculated, and the memory cell having this hash value as address in the hash table **50** is determined (step **S106**). Subsequently, it is determined (namely, retrieved) whether or not there is an OLDB structure which agrees with the IP address of the router that transmitted the received opaque LSA from among one or more OLDB structures connected to the determined memory cell by the pointers (step **S108**). In this retrieval, since a retrieval scope is pinpointed to the group of the OLDB structures corresponding to one memory cell of the hash table **50**, it is possible to retrieve at a high-speed.

[0140] When there is no coincident OLDB structure (NO in step **S118**), the OLDB structure which stored the received opaque LSA is connected to the endmost of the OLDB structure(s) connected to the memory cell determined in step **S106**. For example, when the hash value is hi of **FIG. 12**, the OLDB structure of the received opaque LSA is connected to the rear of the OLDB structure **51***c*, and the pointer to the connected new OLDB structure is stored in the pointer region provided in the OLDB structure **51***c*.

[0141] On the other hand, in step **S108**, when there is any coincident OLDB structure (YES in step **S108**), it is determined whether or not the received opaque LSA is up-to-date (latest temporally) (in other words, the opaque LSA having the up-to-date traffic information) (step **S110**).

[0142] When the received opaque LSA is not up-to-date (NO in step **S110**), it is discarded and the memory region ensured in step **S102** is released (step **S112**). On the other hand, when the received opaque LSA is up-to-date (YES in step **S110**), this up-to-date OLDB structure is exchanged for an old OLDB structure which has already existed in the database **32***b* (step **S114**). For example, when the OLDB structure **51***b* of **FIG. 12** is exchanged by the OLDB structure (referred to as "an OLDB structure x") having the received new opaque LSA, the OLDB structure x is inserted into this same position. Alternatively, the OLDB structure **51***c* is connected to a very rear of the OLDB structure **51***a*, and also the OLDB structure x can be connected to a very rear of the OLDB structure **51***c*.

[0143] Subsequently, the memory region of the OLDB structure of the old opaque LSA is released (step **S116**), and the process is ended.

[0144] One Example of the Load Sharing

[0145] Finally, one example of a load sharing which is carried out on the basis of the traffic information of each router **11** stored in the opaque LSA database **32***b* will be explained as follow.

[0146] Respective logical bandwidths (bandwidths of LSP) [bps] of routes 1, 2, 3 (refer to **FIG. 1**) are set to 10M, 8M, 2M, respectively.

[0147] In order to control the load sharing by means of the TE, the input-side router **11S** (processing unit **30**) first calculates each effective load of the routes 1, 2, 3. Here, "the effective load" is an effective usage rate that is calculated based on a usage rate of the link and a packet discard rate (packet loss rate) in this link. Although the actual load of the link had better be measured, this effective load is used, because it is difficult to measure the actual load directly in the case where the router **11** has a multistage switch configuration.

[0148] When the effective load of route i is set as ρ_effective_path_i, this effective load can be acquired by the following equations (8) and (9), for example:

$$\rho\_effective\_path\_i=\rho\_path\_i \times f(Loss\_path\_i) \quad (8)$$

$$\rho\_effective\_path\_i=\mathrm{Min}\ (\rho\_effective\_path\_i, \rho\_ceiling) \quad (9)$$

[0149] Here, ρ_path_i is an average usage rate of the entire one or more links (link_j) which configures the route i (path_i), and is acquired by the following equation (10):

$$\rho\_path\_i=Average(\mathrm{Ave}\_Utilization(link\_j, path\_i)) \quad (10)$$

[0150] Furthermore, Loss_path_i is the total of a discard packet number (Loss_link_j) of each link which configures the route i, and is acquired by the following equation (11):

$$Loss\_path\_i=\Sigma Loss\_link\_j \quad (11)$$

[0151] The function f is a function for correcting so as to calculate higher a load in the case where the discard of packet is carried out. This is because, when no packet is discarded, the link load ρ_path_i agrees with the effective load, and when the discard of packet is carried out, it is necessary to correct higher the load. ρ_ceiling is an upper limit value of the effective load.

[0152] Now, when the effective loads of the routes 1, 2, 3 are set to ρ_effective_path_1=0.5, ρ_effective_path_2=0.2 and ρ_effective_path_3=0.3, respectively, effective traffic amounts [bps] of the routes 1, 2, 3 are 10M×0.5=5M, 8M×0.2=1.6M and 2M×0.3=0.6M, respectively.

[0153] Subsequently, the router **11S** (processing unit **30**) performs load adjusting. First, all the routes 1, 2, 3 are deemed as a virtual single pipe, and the average usage rate ρave_effective of this pipe is acquired by the following equation:

$$\rho ave\_effective=\Sigma(\rho\_effective\_path\_i \times LBW\_path\_i)+\Sigma LBW\_path\_i \quad (12)$$

[0154] Here, LBW_path_i is a logical bandwidth of the route i.

[0155] When the equation (12) applies the above-mentioned example,

$$\rho ave\_effective=(5M+1.6M+0.6M)/(10M+8M+2M)=0.36$$

[0156] Next, effective bandwidth ΔEBW_path_i [bps] which shifts between the routes is calculated by the following equation:

$$\Delta EBW\_path\_i=(\rho ave\_effective-\rho\_effective\_path\_i)\times LBW\_path\_i \quad (13)$$

[0157] When this is calculated in each of the routes 1, 2, 3, the following answer is obtained:

[0158] The effective bandwidth ΔEBW_path_1, shifting in the route 1=(0.36-0.5)×10M=−1.4M

[0159] The effective bandwidth ΔEBW_path_2, shifting in the route 2=(0.36-0.2)×8M=+1.28M

[0160] The effective bandwidth ΔEBW_path_3, shifting in the route 3=(0.36-0.3)×2M=+0.12M

[0161] The total of the effective bandwidths shifting in the routes is −1.4M+1.28M+0.12M=0

[0162] When the load adjusting is performed based on this calculation result, in the route 1, 5M-1.4M=3.6M, and in the route 2, 1.6M+1.28M=2.88M, and in the route 3, 0.6M+0.12M=0.72M, and in the routes 1, 2, 3, the load sharing (distribution of the traffic) is changed at a ratio of 3.6× Gr:2.88× Gr:0.72× Gr. The symbol Gr denotes a load adjusting coefficient.

[0163] According to the present invention, the nodes in the communication network can collect the information in the traffic in the communication network, and control the load sharing by the information.

What is claimed is:

1. In a communication network having a plurality of nodes, which transmit, receive or transfer communicated information, and a plurality of links, which connect the plurality of nodes to each other, a traffic information collecting device, which is provided in at least one of the plurality of nodes, for collecting traffic information all or some of the plurality of links, comprising:

a traffic information collecting unit for collecting first traffic information of a first link connected to an own node among the plurality of links;

a traffic information transmitting unit for transmitting, to the other nodes, said first traffic information collected by said traffic information collecting unit, using a message prescribed in a communication protocol in the communication network;

a traffic information receiving unit for receiving second traffic information of second links connected to the other nodes among the plurality of links, said second traffic information being transmitted from the other nodes; and

a traffic information storage for storing said first and second traffic information.

2. The traffic information collecting device according to claim 1, wherein said traffic information transmitting unit transmits the first traffic information at predetermined fixed time intervals.

3. The traffic information collecting device according to claim 1, wherein said traffic information transmitting unit transmits the first traffic information when said traffic information receiving unit receives the second traffic information.

4. The traffic information collecting device according to claim 1, wherein said traffic information transmitting unit transmits the first traffic information at predetermined fixed time intervals, as well as when said traffic information receiving unit receives the second traffic information.

5. The traffic information collecting device according to claim 1, further comprising:

a traffic information transferring unit for transferring the second traffic information received by said traffic information receiving unit to the other first links excluding the received first link.

6. The traffic information collecting device according to claim 1, wherein said first and second traffic information is traffic information of a link for outputting the traffic.

7. The traffic information collecting device according to claim 1, wherein said traffic information collecting unit collects the first traffic information at predetermined fixed time intervals.

8. The traffic information collecting device according to claim 1, wherein said first and second traffic information, respectively, includes an average usage rate of the first and second links, the number of a packet discarded in the first and second links, and a bandwidth of the first and second link; and in the case where a logical link is provided in the first and second link, an average usage rate of the logical link, the number of a packet discarded in the logical link, and a bandwidth of the logical link.

9. The traffic information collecting device according to claim 1, wherein said message in the communication protocol is an opaque link state advertisement in the OSPF protocol.

10. The traffic information collecting device according to claim 1, wherein said node is a router.

11. The traffic information collecting device according to claim 1, wherein the node having the traffic information collecting device is a node for controlling a load sharing of the traffic in the communication network.

12. The traffic information collecting device according to claim 1, wherein said traffic information storage comprises a hash table and structure data having the fist or second traffic information of the respective nodes,

said hash table having a memory cell, said memory cell being addressed by a hash value obtained by hashing information for identifying the respective nodes, and storing a pointer to the structure data having the first or second traffic information of the node in correspondence to the hash value of the memory cell, and

said structure data having a pointer to another structure data when there is another structure data in correspondence to the same hash value.

13. The traffic information collecting device according to claim 1, wherein in the case where the first or second traffic information of the same node have already existed in said traffic storage when the first or second traffic information is

stored in same, said traffic information storage stores the first or second traffic information which is later temporally among them.

14. In a communication network having a plurality of nodes, which transmit, receive or transfer communicated information, and a plurality of links, which connect the plurality of nodes to each other, a traffic information collecting device, which is provided in at least one of the plurality of nodes, for collecting traffic information all or some of the plurality of links, comprising:

a traffic information collecting unit for collecting first traffic information of a first link connected to an own node among the plurality of links; and

a traffic information storage for storing said first traffic information collected by the traffic information collecting unit, and second traffic information of second links connected to the other nodes among the plurality of links, said second traffic information being transmitted from the other nodes.

15. A node in a communication network having a plurality of nodes for transmitting, receiving or transferring communicated information, and a plurality of links for connecting the plurality of nodes to each other, comprising:

a traffic information collecting unit for collecting first traffic information of a first link connected to an own node among the plurality of links;

a traffic information transmitting unit for transmitting, to the other nodes, said first traffic information collected by said traffic information collecting unit, using a message prescribed in a communication protocol in the communication network;

a traffic information receiving unit for receiving second traffic information of second links connected to the other nodes among the plurality of links, said second traffic information being transmitted from the other nodes; and

a traffic information storage for storing said first and second traffic information.

16. A node in a communication network having a plurality of nodes for transmitting, receiving or transferring communicated information, and a plurality of links for connecting the plurality of nodes to each other, comprising:

a traffic information collecting unit for collecting first traffic information of a first link connected to an own node among the plurality of links; and

a traffic information storage for storing said traffic information collected by the traffic information collecting unit and second traffic information of second links connected to the other nodes among the plurality of links, said second traffic information being transmitted from the other nodes.

17. In a communication network having a plurality of nodes for transmitting, receiving or transferring communicated information, and a plurality of links for connecting the plurality of nodes to each other, a traffic information collecting method for collecting traffic information of all or some of the plurality of links, and performed by at least one of the plurality of nodes, comprising the steps of:

collecting first traffic information of a first link connected to an own node among the plurality of links;

transmitting, to the other nodes, said first traffic information, using a message prescribed in a communication protocol in the communication network;

receiving second traffic information of second links connected to the other nodes among the plurality of links, said second traffic information being transmitted from the other nodes; and

storing said first and second traffic information in a storage provided in the node.

18. In a communication network having a plurality of nodes for transmitting, receiving or transferring communicated information, and a plurality of links for connecting the plurality of nodes to each other, a traffic information collecting method for collecting traffic information of all or some of the plurality of links, and performed by at least one of the plurality of nodes, comprising the steps of:

collecting first traffic information of a first link connected to an own node among the plurality of links; and

storing, in a storage provided in the node, said first traffic information, and second traffic information of second links connected to the other nodes among the plurality of links, said second traffic information being transmitted from the other nodes.

19. In a communication network having a plurality of nodes for transmitting, receiving or transferring communicated information, and a plurality of links for connecting the plurality of nodes to each other, a traffic information collecting program product for collecting traffic information of all or some of the plurality links, and executed by at least one of the plurality of nodes, comprising:

a collecting process for collecting first traffic information of a link connected to an own node among the plurality of links;

a transmitting process of transmitting, to the other nodes, said first traffic information, using a message prescribed in a communication protocol in the communication network;

a receiving process for receiving second traffic information of second link connected to the other nodes among the plurality of links, said second traffic information being transmitted from the other nodes; and

a storing process for storing said first and second traffic information in a storage provided in the node.

20. In a communication network having a plurality of nodes for transmitting, receiving or transferring communicated information, and a plurality of links for connecting the plurality of nodes to each other, a traffic information collecting program product for collecting traffic information of all or some of the links, and executed by at least one of the plurality of nodes, comprising:

a collecting process for collecting first traffic information of a first link connected to an own node among the plurality of links; and

a storing process for storing, in a storage provided in the node, said first traffic information, and second traffic information of second links connected to the other nodes among a plurality of links, said second traffic information being transmitted from the other nodes.

* * * * *

US 20040010583A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2004/0010583 A1

Yu et al. (43) Pub. Date: Jan. 15, 2004

(54) METHOD AND APPARATUS FOR DEFINING FAILOVER EVENTS IN A NETWORK DEVICE

(75) Inventors: **Ken Yu**, Burlington, MA (US); **Ramasamy Jesuraj**, Westford, MA (US); **Arun Kudur**, Billerica, MA (US); **Shang Chang**, Medford, MA (US)

Correspondence Address:
JOHN C. GORECKI, ESQ.
165 HARVARD ST.
NEWTON, MA 02460 (US)

(73) Assignee: **Nortel Networks Limited**, St. Laurent (CA)

(21) Appl. No.: **10/192,663**

(22) Filed: **Jul. 10, 2002**

Publication Classification

(51) **Int. Cl.**[7] **G06F 15/16**

(52) **U.S. Cl.** **709/224**; 709/230

(57) **ABSTRACT**

A critical interface may be defined for a network device such that if the critical interface goes DOWN, the network device will perform a forced failover. IP addresses are assigned to interface groups, and interface groups are assigned to a critical interface. The critical interface will go DOWN if any one of the interface groups goes DOWN. The interface groups will not go DOWN, however, unless all members' IP addresses assigned to the interface group go DOWN. By configuring the critical interface in this manner, the network manager has increased flexibility in defining which events should and should not trigger failover of the network device. Additionally, combinations of events may be grouped to enable the network manager to take into account fairly complex failure scenarios and specify, with precision, the action to be taken by the network device under myriad possible situations.

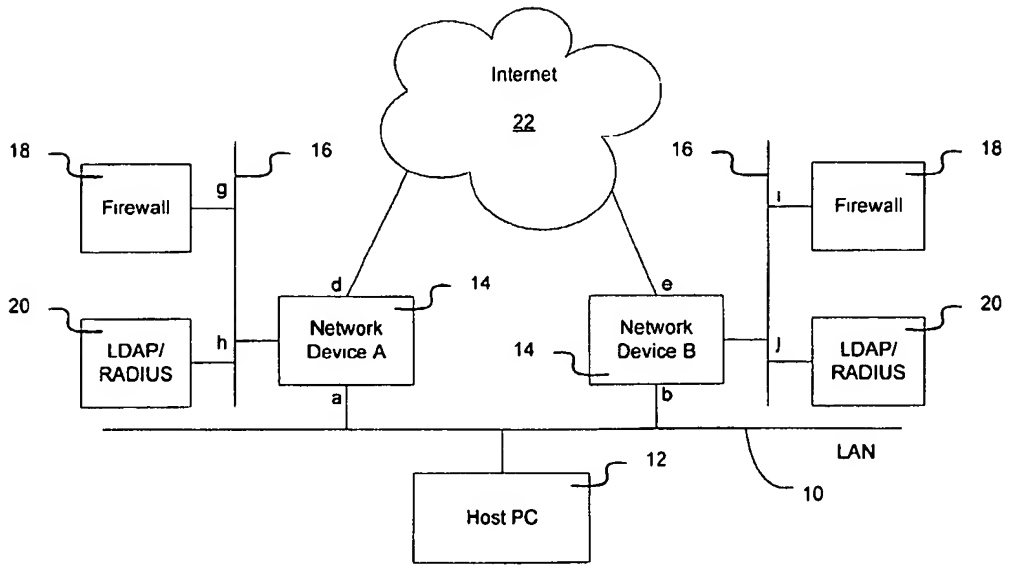


Figure 1

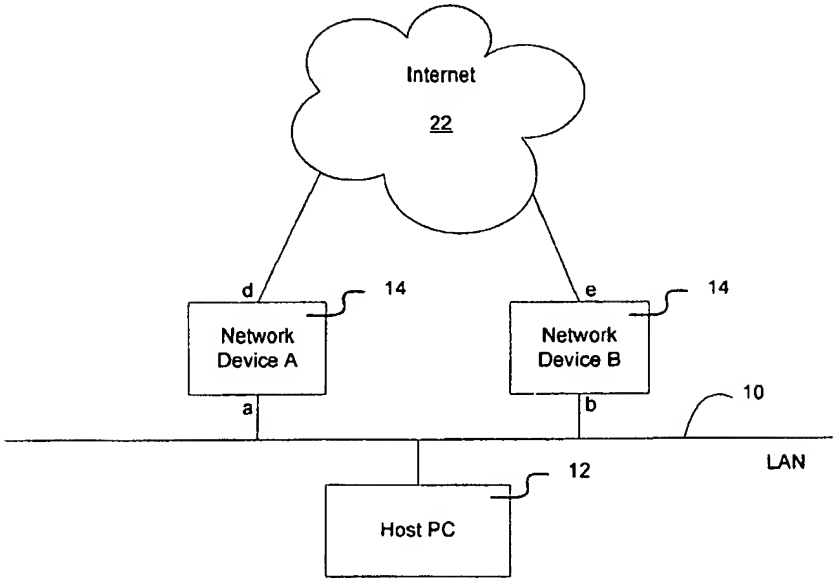


Figure 2

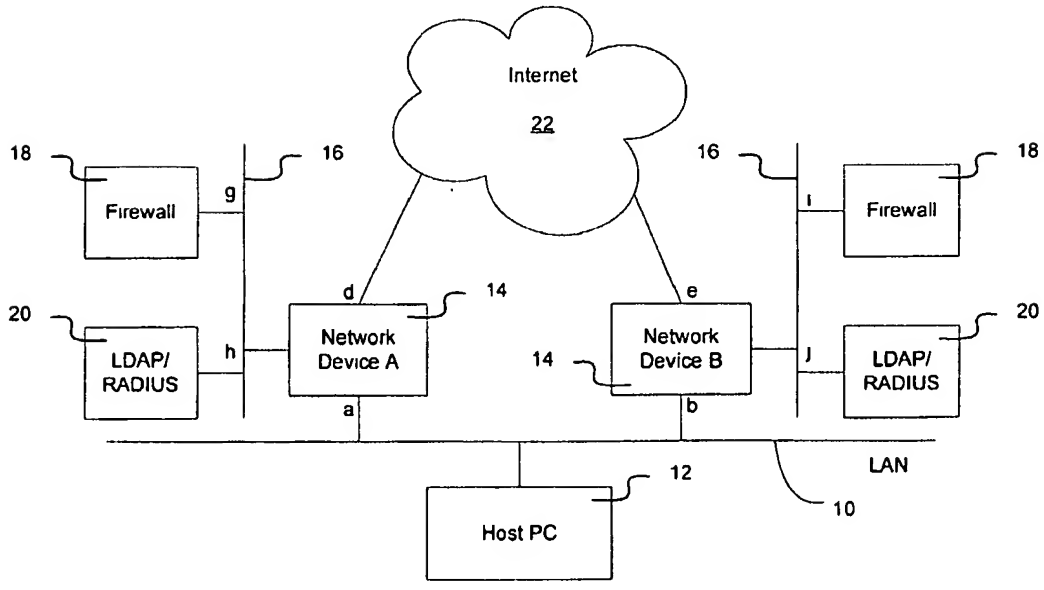


Figure 3

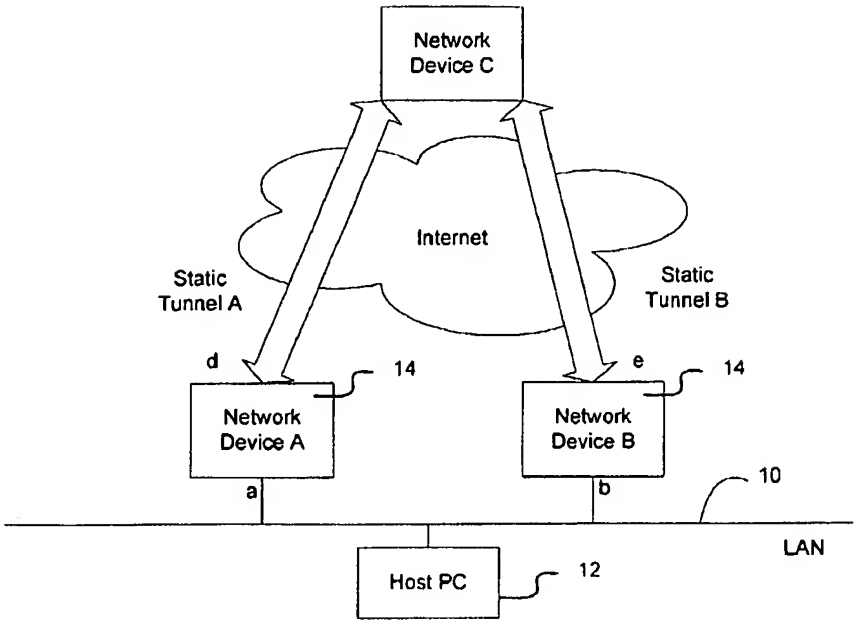


Figure 4

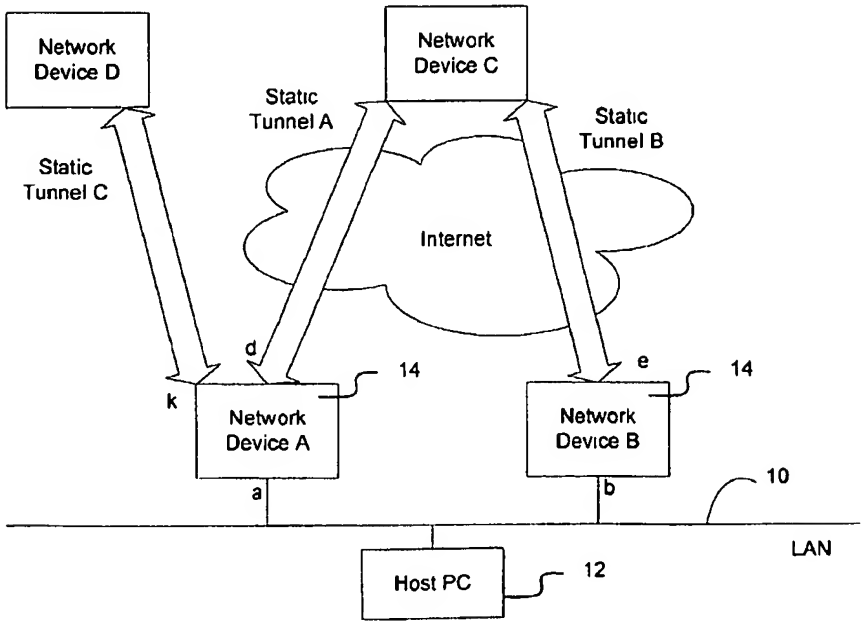


Figure 5

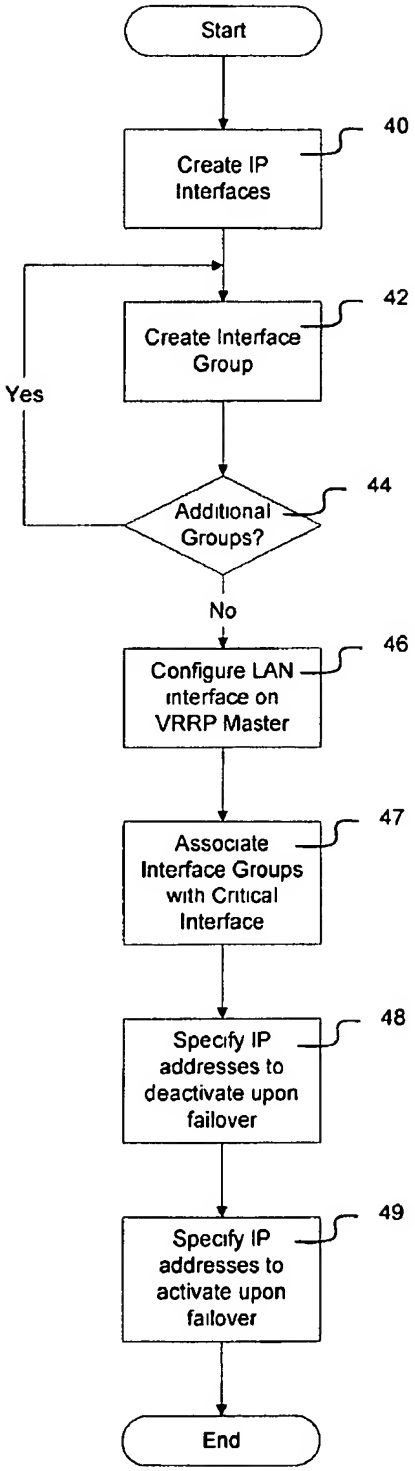

Start
Create IP Interfaces
40
Create Interface Group
42
Yes
Additional Groups?
44
No
Configure LAN interface on VRRP Master
46
Associate Interface Groups with Critical Interface
47
Specify IP addresses to deactivate upon failover
48
Specify IP addresses to activate upon failover
49
End

Figure 6

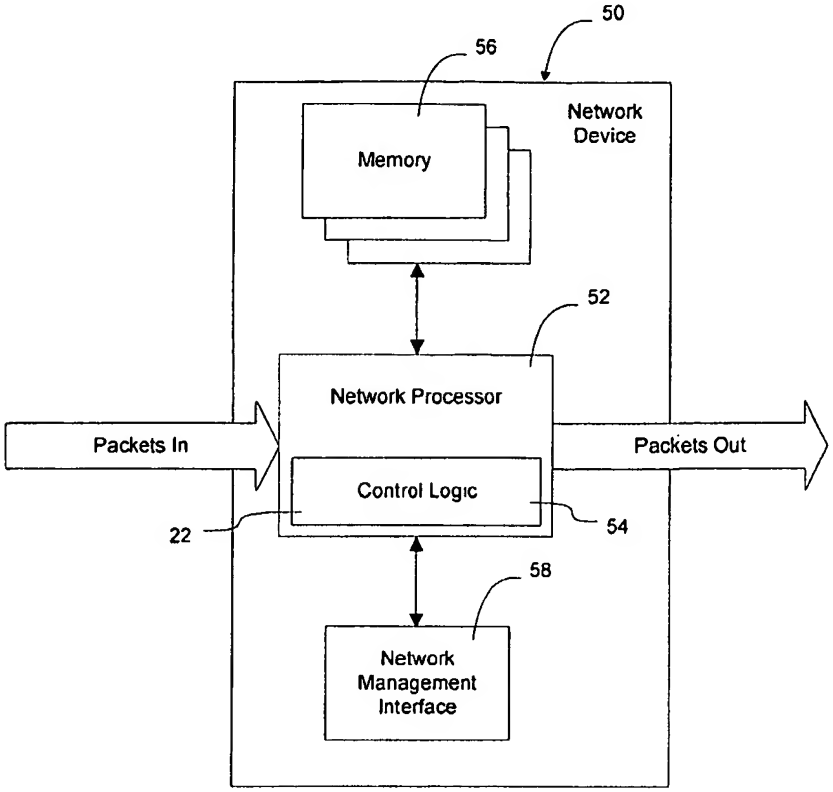
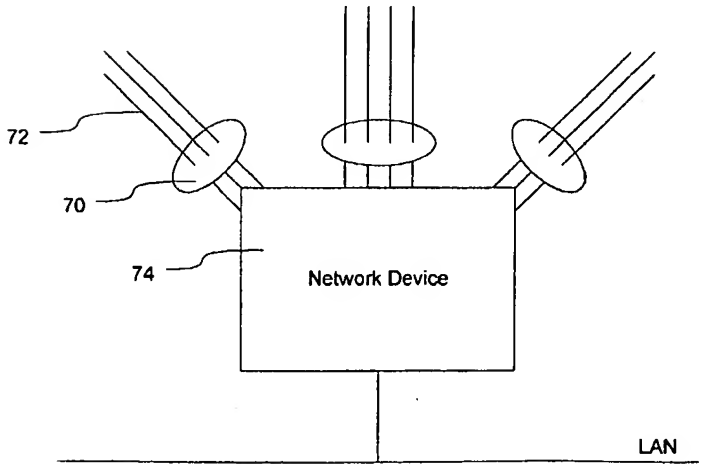


Figure 7



METHOD AND APPARATUS FOR DEFINING FAILOVER EVENTS IN A NETWORK DEVICE

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates to communication networks and, more particularly, to a method and apparatus for defining failover events in a network device.

[0003] 2. Description of the Related Art

[0004] Data communication networks may include various computers, servers, nodes, routers, switches, hubs, proxies, and other network devices coupled to and configured to pass data to one another. These various network elements will be referred to herein as "network devices." Data is communicated through the data communication network by passing data packets (or data cells or segments) between the network devices by utilizing one or more communication links between the devices. A particular packet may be handled by multiple network devices and cross multiple communication links as it travels between its source and its destination over the network.

[0005] A particular physical network device may be logically partitioned into multiple logical network devices to facilitate network management and increase the number and types of services offered by that network device. For example, a router may be partitioned into multiple virtual routers, each of which is a collection of threads, either static or dynamic, that provides routing and forwarding services much like physical routers. A virtual router need not be a separate operating system process (although it could be); it simply has to provide the illusion that a dedicated router is available to satisfy the needs of the network(s) to which it is connected.

[0006] As data networks have grown in complexity and speed, the network devices used in those networks have likewise increased in complexity and speed. Unfortunately, this complexity provides manifest opportunities for the network to fail. To increase the reliability of the network, networks are designed such that in the event of a failure of a network device or link, an alternate network device or link may be used until a repair can be made to the failed element. This notion will be referred to herein as "redundancy."

[0007] As services offered over networks become increasingly complex, for example layer 2 and layer 3 virtual private networks (VPNs) are deployed, and virtual private LAN segment (VPLS) services are made available, redundancy becomes increasingly important, and also increasingly difficult. Specifically, the redundant network devices must share information as to the types of tunnels, etc. that are being set up, so that in the event of failure of one network device another network device can continue to offer these specialized services. The process undertaken by the network device(s) to exchange responsibility for provision of services will be referred to herein as failover.

[0008] The various network devices on the communications network communicate with each other using predefined sets of rules, referred to herein as protocols. Different protocols are used to govern different aspects of the communication, such as how signals should be formed for transmission between network devices, various aspects of what the data packets should look like, and how packets should be handled by the network devices.

[0009] One particular protocol, known as Virtual Router Redundancy Protocol (VRRP), specifies an election protocol for use in a broadcast domain, such as an Ethernet domain, that dynamically assigns responsibility for a virtual router to one of the physical VRRP routers on a local area network. This allows any of the virtual router IP (internet protocol) addresses on the local area network (LAN) to be used as the default first hop router by end-hosts. VRRP is described in greater detail in Internet Engineering Task Force (IETF) Request For Comments (RFC) 2338, the content of which is hereby incorporated herein by reference in its entirety.

[0010] VRRP normally permits two or more routers to share state information so that one of the routers can function as a "hot standby" for the other. It is combined with a "keep alive" mechanism such that when the standby router detects a failure of the primary, it is able to then impersonate the primary router during failover. One advantage gained by using VRRP is a higher availability default party without requiring configuration of dynamic routing or router discovery protocols on the end hosts.

[0011] The events that trigger failover are limited in VRRP. Specifically, VRRP is designed to deal with a single point of failure and only provides failover of a particular router or virtual router when that particular router is experiencing failure. Unfortunately, this does not account for other conditions under which a network manager may desire to effect failover.

[0012] One conventional system (the Bay RS available from Nortel Networks) attempted to rectify some of the shortcomings of VRRP. Specifically, this system enabled the network manager to effect failover of the master in the event a designated IP address went down. While this provides a good first step to enabling failover control, it does not enable a network manager to account for the myriad possible failures in current and envisioned networks.

SUMMARY OF THE INVENTION

[0013] The present invention overcomes these and other drawbacks by providing a method and apparatus through which an user may define failover events in a network device by establishing a critical interface for the network device, such that if the critical interface goes down, the network device will perform failover. In one embodiment, a network manager initially assigns IP addresses to interface groups, and interface groups are then assigned to the critical interface. The critical interface, in this embodiment, will go down if any one of the interface groups goes down. The interface groups will not go down, however, unless all members' IP addresses assigned to the interface group go down. By configuring the critical interface in this manner, the network manager has increased flexibility in defining which events should and should not trigger failover of the network device. Additionally, combinations of events may be grouped to enable the network manager to take into account fairly complex failure scenarios and specify, with precision, the action to be taken by the network device under myriad possible situations.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] Aspects of the present invention are pointed out with particularity in the appended claims. The present inven-

tion is illustrated by way of example in the following drawings in which like references indicate similar elements. The following drawings disclose various embodiments of the present invention for purposes of illustration only and are not intended to limit the scope of the invention. For purposes of clarity, not every component may be labeled in every figure. In the figures:

[0015] FIGS. **1-4** are functional block diagrams of example networks configured to implement embodiments of the invention;

[0016] **FIG. 5** is a flowchart of an example of software that can be used to implement one embodiment of the invention;

[0017] **FIG. 6** is a functional block diagram of a network device according to an embodiment of the invention; and

[0018] **FIG. 7** is a functional block diagram of a network device illustrating interfaces grouped into interface groups.

DETAILED DESCRIPTION

[0019] The following detailed description sets forth numerous specific details to provide a thorough understanding of the invention. However, those skilled in the art will appreciate that the invention may be practiced without these specific details. In other instances, well-known methods, procedures, components, protocols, algorithms, and circuits have not been described in detail so as not to obscure the invention.

[0020] As described in greater detail below, the invention enables an user to define a critical interface for each network device such that if the critical interface goes down, the network device will perform failover. In this embodiment, IP addresses are first assigned to interface groups, and interface groups are then assigned to the critical interface. The critical interface will go down if any one of the interface groups goes down. The interface groups will not go down, however, unless all members' IP addresses assigned to the interface group go down. By configuring the critical interface in this manner, the network manager has increased flexibility in defining which events should and should not trigger failover of the network device. Additionally, combinations of events may be grouped to enable the network manager to take into account fairly complex failure scenarios and specify, with precision, the action to be taken by the network device under myriad possible situations.

[0021] Interface Groups and the Critical Interface

[0022] An Interface Group is a group of interfaces selected by the network manager to collectively define when one or more network device should perform failover. The interface group may include one or more physical interfaces, one or more IP addresses, one or more tunnel interfaces, or any combination thereof. For example, the members of an interface group may include the IP address of Firewall, LDAP/RADIUS server, branch office tunnel, VPN tunnel, LAN/WAN physical interface, and an IP interface (e.g. Virtual circuit in ATM/FR) supported by the system, although the invention is not limited to these illustrative examples. The formation of interface groups **70** from individual interfaces **72** to the network device **74** is illustrated in **FIG. 7**.

[0023] An Interface Group has three status: Up, Down, and Quasi. In one embodiment of the invention, when all the member interfaces are up, the status is UP; when all the interfaces are down, the status is DOWN; when at least one interface is down, and at least one interface is up, the status is QUASI. Other embodiments may employ different conventions and the invention is not limited to this particular embodiment.

[0024] Interface groups are not limited to IP interfaces on a particular VRRP master. Rather, an interface group may be associated with more than one VRRP master interface. Likewise, an interface group may be associated with more than one private physical interface running VRRP. Thus if the interface group goes down, all associated master interfaces will perform failover.

[0025] The network device manager may assign a name to the interface group to enable the interface group to be referenced at a later time. In one embodiment, the network manager may also add additional or delete existing IP interfaces from the interface group once it has been established, although the invention is not limited to network devices employing this feature.

[0026] A physical interface on which VRRP has been configured to run as master is called a VRRP master. According to one embodiment of the invention, a LAN interface that is running as a VRRP master is associated with a critical interface. The critical interface, in this embodiment, is a collection of one or more Interface Groups. When one or more Interface Groups in the Critical Interface goes DOWN, the VRRP master will perform a VRRP-like failover and cause mastership to transfer to the VRRP standby network device. Specifically, when any one of the interface groups goes down, the VRRP master sends a packet with priority 0 advertising to the VRRP standby to assume mastership of the virtual IP address. Alternatively, the VRRP master may keep quiet and discontinue sending out keep alive messages, in which case mastership of the virtual IP address will be timed out after a predetermined period. In either event, mastership of the virtual IP address will be transferred from the master to the standby network device. The VRRP master interface will stay in the DOWN state until all associate interface groups have come UP, and then will reclaim mastership of the virtual IP address. Since an interface group in the QUASI state is not DOWN, the Critical Interface will treat an Interface Group in QUASI state as UP.

[0027] VRRP is a state machine protocol that specifies three states for the LAN interface: Initialization, Backup, and Master. According to one embodiment of the invention, a fourth state, Critical, is added to the conventional VRRP states. Critical state, in this context, means that at least one of the interface groups associated with the Critical Interface is DOWN. This enables the network device to distinguish between an actual failure of the physical interface and a "faked" failure associated with the Critical Interface going DOWN.

[0028] The VRRP master will stay in the Critical state until all the associated interface groups are UP, or are administratively disabled or removed. When all interface groups associated with the critical interface return to UP status, the system reclaims mastership of the VRRP interface.

[0029] During initialization state, if a virtual router becomes the Master for an interface, it checks if there are

associated interface groups associated with its critical interface. If there are, the virtual router then checks the status of all interface groups associated with the critical interface. If the status of all of the associated interface groups is UP, then the virtual router assumes mastership of the interface. Otherwise, the virtual router becomes the backup virtual router and remains in that state until the status of all interface groups associated with the critical interface is UP.

[0030] The critical interface, according to one embodiment of the invention, is only enabled where the physical interface is configured as a VRRP master, and is not enabled where the physical interface is configured as a VRRP backup, even where the VRRP backup has assumed mastership. By configuring the physical interfaces in this manner the backup is more robust and less likely to attempt failover where there isn't another network device to assume mastership from the backup network device.

[0031] Interface Group and Critical Interface Examples

[0032] **FIG. 1** illustrates an example in which a local area network LAN **10** is configured to interconnect a host PC **12** and two network devices **14**, designated in **FIG. 1** as network devices A and B. In this example, VRRP is configured on the private physical interface "a" on network device A, and on private physical interface "b" on network device B. Network device A and network device B are configured to back each other up in this network with network device A as the master and network device B as the standby.

[0033] Assume, for this example, that tunnel interfaces "d" and "e" connect network devices A and B to a remote system (not illustrated) across the Internet **22**. To effect automatic fail over between network device A and network device B, it has been conventionally necessary to run a dynamic routing protocol, such as OSPF, on the private network **10** and on the tunnel interfaces "d" and "e." Unfortunately, many network managers are reluctant to use dynamic routing on the LAN just to effect failover, since it is difficult to provide secure tunnels through the Internet using dynamic routing.

[0034] By defining a VRRP critical interface, as described in greater detail above, it is possible to effect failover upon failure of a tunnel. Specifically, in this example, the network manager would first define an interface group containing the tunnel interface "d." The network manager would then assign this interface group to a critical interface associated with the private physical interface "a" of network device A. Thus, if the tunnel goes down, the interface group is down, and the critical interface associated with network device A will be down. This will cause network device A to instruct network device B to assume mastership of the virtual IP address associated with the private physical interface "a" of network device A, which will, in turn, cause network device B to assume mastership for that interface. Thus, tunnel fail over may be achieved without running a dynamic routing protocol.

[0035] Another example is illustrated in **FIG. 2**. In this example, as in the example illustrated in **FIG. 2**, a local area network LAN **10** is configured to interconnect a host PC **12** and two network devices **14**, designated in **FIG. 2** as network devices A and B. In this example, VRRP is configured on the private physical interface "a" on network device A, and on private physical interface "b" on network device B. Network device A and network device B are configured to back each other up in this network with network device A as the master and network device B as the standby. In this example, each network device is further configured to be connected to a secondary LAN **16** interconnecting the network device with a firewall **18** and LDAP/RADIUS server **20**.

[0036] A server hosting a LDAP directory will be referred to herein as an LDAP server. LDAP (Lightweight Directory Access Protocol) is a client-server protocol for accessing a directory service. LDAP allows a user to locate organizations, individuals, and other resources such as files and devices in a network whether or not the user knows the domain name, IP address, or geographic whereabouts of the resource. An LDAP directory can be distributed among many servers on a network, then replicated and synchronized regularly.

[0037] RADIUS (Remote Authentication Dial-In User Service) is a client/server protocol and software that enables remote access servers to communicate with a central server to authenticate dial-in users and authorize their access to the requested system or service. RADIUS allows a company to maintain user profiles in a central database that all remote servers can share. It provides better security, allowing a company to set up a policy that can be applied at a single administered network point.

[0038] Assume for this example that the network manager would like to effect failover if the tunnel interface "d" is down, or if the firewall and the LDAP/RADIUS server are down. In this example, the network manager would first define a first interface group containing the tunnel interface "d" and then define a second interface group containing the firewall interface "g" and LDAP/RADIUS server interface "h." Optionally, the network manager could also define a third interface group containing the interface to the LAN **16**. The network manager would then assign these interface groups to the critical interface associated with the private physical interface "a" on network device A.

[0039] Since all members of an interface group must be down to cause the interface group to assume DOWN status, a failure at tunnel interface "d" will cause the tunnel interface group to go DOWN. Similarly, a failure of both the firewall interface "g" and the LDAP/RADIUS interface "h" will cause that associated interface group to assume a DOWN status. A failure of either the firewall interface "g" or the LDAP/RADIUS interface "h," by itself, however, will not cause that associated interface group to assume a DOWN status. If either interface group goes DOWN, the critical interface will be DOWN, and mastership will pass to network device B. Accordingly, the network manager of the network device has great flexibility in determining what combinations of events should cause the network device to failover and what combinations of events should be tolerated by the network device.

[0040] Customized Response to Perceived Failure on the Network

[0041] As described above, when a critical interface goes down, the network device performs a forced failover to cause mastership to pass to standby network device. Situations may arise, however, where it is instead desirable to take

other action or, optionally, to take other actions in addition to switching mastership to the standby network device. For example, in certain situations, it may be advantageous to enable the network device to respond to perceived network failures by selectively enabling or disabling one or more interfaces or interface groups.

[0042] One example of where it may be advantageous to disable an interface to the network device is illustrated in FIG. 3. In the example illustrated in FIG. 3, network device A is configured as the VRRP master and network device B is configured as the VRRP standby. Two tunnels, static tunnel A and static tunnel B, are connected between network devices A and C, and network devices B and C, respectively. Static tunnel A, in this example, has a cost of 0 and static tunnel B has a cost of 2. Because of the cost of the two static tunnels, static tunnel A will become the primary tunnel and static tunnel B will become the secondary tunnel, and traffic will flow through static tunnel A.

[0043] If the critical interface to network device A causes network device A to failover to network device B, network device B will assume mastership of the virtual IP address on the LAN 10. However, network device C does not know that traffic should be rerouted to the secondary static tunnel because the tunnel interface "d" on network device A is still operational. Thus, traffic will continue to arrive from network device C at tunnel interface "d" on network device A.

[0044] According to one embodiment of the invention, by enabling the network manager to specify that the interface d should be deactivated upon critical interface failover, the network device is provided with the ability to disable the static tunnel to thereby cause traffic to be automatically rerouted to the alternate static tunnel. In this embodiment, the network device may disable the interface d by either timing out, or more preferably, by communicating with the network device C to instruct network device C that the tunnel is being shut down.

[0045] By deactivating the interface during the failover procedure, using the inherent static tunnel failover routing protocol features, the network device C will recognize that the static tunnel has been shut down and will automatically reroute all user traffic to the secondary static tunnel connected to network device B. Later, when network device A regains mastership, the deactivated member in the critical interface will be reactivated. Optionally, interface e on network device B may likewise be deactivated when mastership is restored to network device A, to cause network device C to route traffic back through the primary tunnel. Thus, tunnel failover can be achieved without running a dynamic routing protocol.

[0046] A network device may also find it advantageous to selectively enable another interface upon VRRP failover. For example, as illustrated in FIG. 4, a network device with a static tunnel to network device C may wish to open a static tunnel to network device D upon VRRP failover. This may be desirable in any number of situations, such as to enable continued communication through the network via a different tunnel or to enable a network manager to assess and repair the perceived failure. Accordingly, according to another embodiment of the invention, the network manager is provided with the ability to selectively enable interfaces upon critical interface failover.

[0047] Although deactivation or activation of one or more interfaces may be of particular use in connection with handling static tunnels, it may also be more broadly applicable to other IP addresses as well. Thus, when a critical interface is DOWN, the network manager may elect to activate or deactivate one or more interface groups or members of an interface group. Enabling selective activation or deactivation in the event of forced failover provides the network manager enhanced flexibility in handling failure scenarios in the network.

[0048] In operation, during the failover procedure for primary IP address failure or for forced failover, all interfaces to the network device or, optionally, all members of each of the critical interface groups, will be scanned to determine if there is an activation or deactivation indication. If the deactivation indicator is found, the IP address will be used to locate the tunnel and the tunnel will be disabled. If the IP address is an interface IP address, it will be disabled. Similarly, if the enable indicator is found, the IP address will be enabled and the network device will continue to communicate via that interface even though mastership of the virtual IP address will pass to the standby network device through failover procedures.

[0049] Implementation

[0050] FIG. 5 illustrates a flowchart of an example of software that can be used to implement an embodiment of the invention. As shown in FIG. 5, the network manager will first create the IP interfaces that need to be covered in the failover scenario (40). Once all IP interfaces have been created, the network manager will create an Interface Group (42), give a name to that group, and select the IP interfaces from the list of IP interfaces that are to become members of that group. Typically these members are the interfaces that needed to be backed up upon failure or that are important to operation of, or services provided by, the network device. If desired, more than one Interface Group may be created for a given VRRP master (44), although a network device manager may decide, for practical considerations, to limit the number of interface groups (e.g. a maximum of three groups) that may be associated with a given particular VRRP master.

[0051] Once the IP interfaces and Interface Groups have been defined, the LAN interface on the VRRP master that will cause the failover is configured (46). The LAN interface of the VRRP master should not be a member of any of the interface groups associated with it, however, since VRRP will cause failover of the VRRP master if the physical interface fails. Accordingly, defining the LAN interface as a member of an interface group may cause the VRRP master to failover according to VRRP, as well as perform a forced failover according to concepts described herein. This may cause confusion and, for practical reasons, should be avoided.

[0052] Interface groups are then assigned to the critical interface for the VRRP master (47). In operation, the network device will use the critical interface to effect failover as directed by the network manager. Optionally, as discussed in greater detail above, the network manager may specify one or more of the IP addresses to be deactivated upon failover (48), or to be activated upon failover (49).

[0053] One example of a network device that may be used in connection with the various embodiments of this invention is illustrated in FIG. 6. As shown in FIG. 6, a network

device **50** configured to receive packets and output packets includes, in this embodiment, a network processor **52** with control logic **54** configured to implement the functions described in greater detail above. A memory **56**, internal to the network device as shown, or external to the network device, may be provided to store computer instructions to enable the network device to perform the functions ascribed to it herein. A physical or virtual network management interface **58** may be provided to enable the network manager to interact with the network device.

[0054] The control logic **54** of the network device may be implemented as a set of program instructions that are stored in a computer readable memory **56** and executed on a microprocessor, such as network processor **52**. However, it will be apparent to a skilled artisan that all logic described herein can be embodied using discrete components, integrated circuitry, programmable logic used in conjunction with a programmable logic device such as a Field Programmable Gate Array (FPGA) or microprocessor, or any other device including any combination thereof. Programmable logic can be fixed temporarily or permanently in a tangible medium such as a read-only memory chip, a computer memory, a disk, or other storage medium. Programmable logic can also be fixed in a computer data signal embodied in a carrier wave, allowing the programmable logic to be transmitted over an interface such as a computer bus or communication network. All such embodiments are intended to fall within the scope of the present invention.

[0055] It should be understood that various changes and modifications of the embodiments shown in the drawings and described in the specification may be made within the spirit and scope of the present invention. Accordingly, it is intended that all matter contained in the above description and shown in the accompanying drawings be interpreted in an illustrative and not in a limiting sense. The invention is limited only as defined in the following claims and the equivalents thereto.

What is claimed is:

1. A method of forcing failover of a network device, comprising the steps of:

establishing at least one interface group;

assigning one or more of said at least one interface groups to a critical interface; and

causing the network device to failover upon a change in state of the critical interface.

2. The method of claim 1, wherein the step of establishing at least one interface group comprises selecting, by a network manager, at least one member from:

one or more physical interfaces,

one or more IP addresses, and

one or more tunnel interfaces.

3. The method of claim 1, wherein the step of establishing at least one interface group comprises selecting, by a network manager, at least two members from:

one or more physical interfaces,

one or more IP addresses, and

one or more tunnel interfaces, and

defining the at least two members as the interface group.

4. The method of claim 2, wherein one of the at least one member is one of a tunnel interface, a firewall interface, and an LDAP/RADIUS interface.

5. The method of claim 2, wherein the interface group may be associated with more than one network device.

6. The method of claim 2, wherein the interface group may be assigned a name by the network manager.

7. The method of claim 1, wherein the interface group has a status, and wherein a change in status will cause the change in state of the critical interface.

8. The method of claim 2,

wherein the interface group has a status,

wherein a change in status will cause the change in state of the critical interface; and

wherein the status of the interface group is selected from at least one of UP, DOWN, and QUASI.

9. The method of claim 8, wherein the status of the interface group is

UP when all members of the interface group are UP,

DOWN when all members of the interface group are DOWN, and

QUASI when at least one member of the interface group is UP and at least one member of the interface group is DOWN.

10. A network device, comprising control logic configured to enable a network manager to:

establish at least one interface group; and

assign one or more of said at least one interface groups to a critical interface; and

wherein the network device is configured to failover upon a change in state of the critical interface.

11. The network device of claim 10, wherein the control logic is configured to enable the network manager to select at least one member for inclusion in said at least one interface group from available interfaces comprising:

one or more physical interfaces,

one or more IP addresses, and

one or more tunnel interfaces.

12. The network device of claim 10, wherein the control logic is configured to enable the network manager to select at least two members for inclusion in said at least one interface group from available interfaces comprising:

one or more physical interfaces,

one or more IP addresses, and

one or more tunnel interfaces; and

wherein the control logic is configured to enable the network manager to define the at least two members as the interface group.

13. The network device of claim 11, wherein one of the at least one member is one of a tunnel interface, a firewall interface, and an LDAP/RADIUS interface.

14. The network device of claim 11, wherein the interface group may be associated with more than one network device.

15. The network device of claim 11, wherein the interface group may be assigned a name by the network manager.

16. The network device of claim 10, wherein the interface group has a status, and wherein a change in status will cause the change in state of the critical interface.

17. The network device of claim 11,

wherein the interface group has a status,

wherein a change in status will cause the change in state of the critical interface; and

wherein the status of the interface group is selected from at least one of UP, DOWN, and QUASI.

18. The network device of claim 17, wherein the status of the interface group is

UP when all members of the interface group are UP,

DOWN when all members of the interface group are DOWN, and

QUASI when at least one member of the interface group is UP and at least one member of the interface group is DOWN.

* * * * *

US 20040085965A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2004/0085965 A1

Fotedar (43) Pub. Date: May 6, 2004

(54) REDUNDANT ROUTER NETWORK

(76) Inventor: Shivi Fotedar, San Jose, CA (US)

Correspondence Address:
MARGER JOHNSON & MCCOLLOM PC
1030 SW MORRISON STREET
PORTLAND, OR 97205 (US)

(21) Appl. No.: 10/286,957

(22) Filed: Oct. 31, 2002

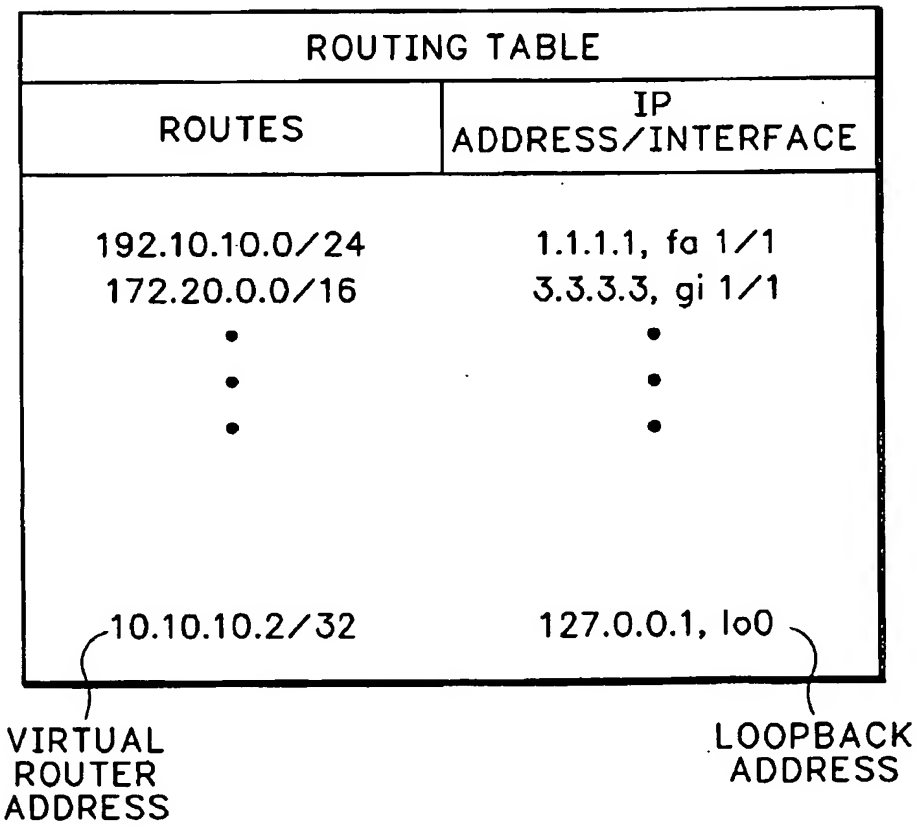
Publication Classification

(51) Int. Cl.[7] H04L 12/28

(52) U.S. Cl. 370/395.31; 370/397

(57) ABSTRACT

A networking system including virtually addressed devices is described. A network device has one or more virtual addresses assigned to its loopback interface. Once so assigned and the loopback interface enabled, data and commands directed to any of the assigned virtual addresses is sent through the device's protocol stack. The device is then structured to act on the data and commands destined for the virtual address.



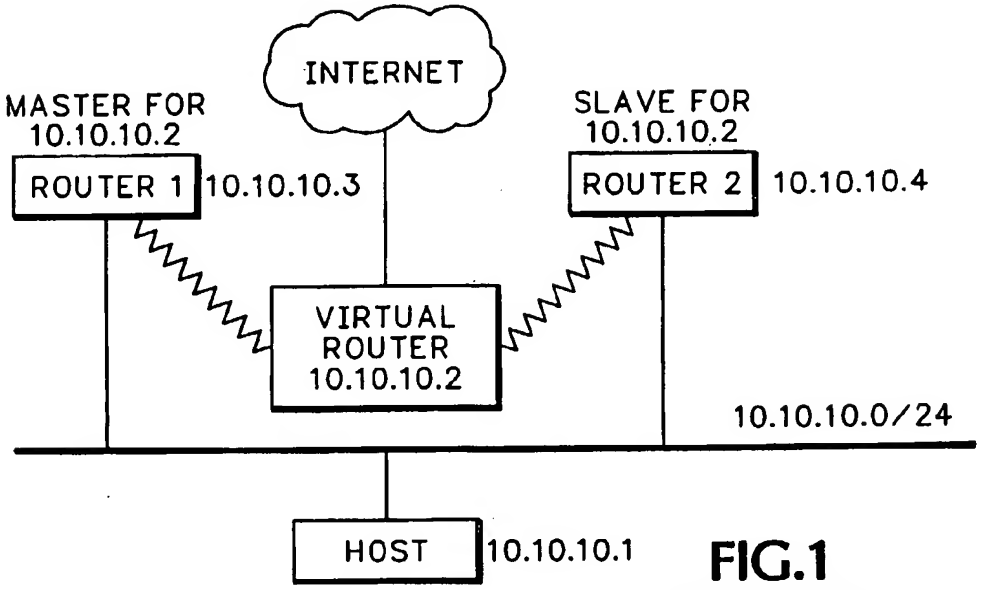


FIG.1
(PRIOR ART)

ROUTING TABLE	
ROUTES	IP ADDRESS/INTERFACE
192.10.10.0/24	1.1.1.1, fa 1/1
172.20.0.0/16	3.3.3.3, gi 1/1
•	•
•	•
•	•
10.10.10.2/32	127.0.0.1, lo0

VIRTUAL ROUTER ADDRESS

LOOPBACK ADDRESS

FIG.2

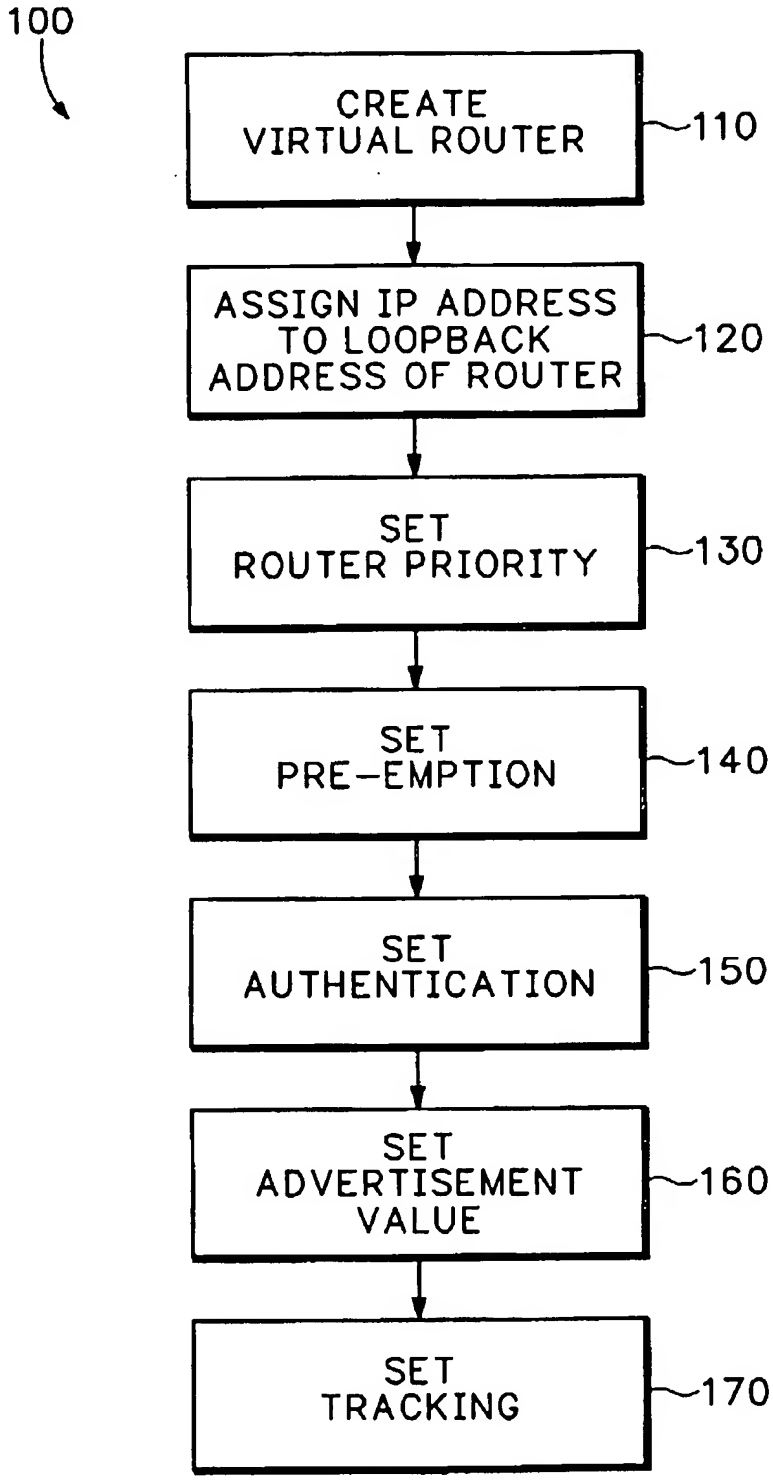

100
CREATE VIRTUAL ROUTER
110
ASSIGN IP ADDRESS TO LOOPBACK ADDRESS OF ROUTER
120
SET ROUTER PRIORITY
130
SET PRE-EMPTION
140
SET AUTHENTICATION
150
SET ADVERTISEMENT VALUE
160
SET TRACKING
170

FIG.3

REDUNDANT ROUTER NETWORK

TECHNICAL FIELD

[0001] This disclosure relates to a system for network processing, and, more specifically, to a network system that can interact directly with redundant routers using virtual addressing.

BACKGROUND

[0002] A router is a network device that connects a host computer on a first network to a computer on a second, similar type network. Previously, routers on Internet Protocol (IP) networks had static physical addresses, and the host would send data packets to the static addresses of the router to be forwarded to the destination network. It was inconvenient when routers failed, because the host would either have to be re-programmed to send data to a different router, or a new router would have to be programmed to have the same static address of the old router. Doing either of these tasks was relatively time consuming and, until complete, the host could not communicate with the desired second network, thus interrupting data traffic between the host and the second network.

[0003] Virtual routing was developed to ameliorate the above problem. In virtual routing, two or more physical routers provide support for a single "virtual" router address. One physical router is deemed the master, and has primary responsibility for routing data sent to a virtual router address. If for some reason the master router should fail, a slave, which is a second physical router, automatically takes over responsibility for routing the data sent to the virtual router address.

[0004] For example, in **FIG. 1** a host computer is assigned IP address 10.10.10.1 and is located on a local subnet 10.10.10.0/24. If the host computer desires to communicate with a computer not on the local subnet, it must send data packets through a router. In **FIG. 1**, a virtual router is assigned the IP address 10.10.10.2. A router **1** is the master for the virtual router, and has a physical address of 10.10.10.3. A router **2** is the slave for the virtual router and has the physical address 10.10.10.4. In normal operation, router **1** performs the actual data routing for the virtual router by routing data sent by the host to the virtual router address 10.10.10.2. If, for some reason router **1** becomes inoperative, router **2** automatically takes over the routing responsibilities of the virtual router from the disabled router **1**. When the slave is active, it routes the data sent by the host to the virtual address 10.10.10.2. Note that even when different physical routers handled the routing functions, the host still sent data to 10.10.10.2, and did not have to be modified when the primary router went down. A protocol, Virtual Router Redundancy Protocol (VRRP), is responsible for ensuring that the master/slave relation and switchovers occur without problem.

[0005] Although the VRRP switching is a relatively robust technology, it creates another problem in network communications. Specifically, although the host computer can send data through a virtual router using the virtual address, it cannot send data or commands to the virtual router address directly. Instead, to send data or commands to the router presently posing as the virtual router, the host must know the physical address of the router currently acting as the virtual router. Thus, with reference to **FIG. 1**, the host could send a ping or some other command to either of the physical routers 10.10.10.3 or 10.10.10.4, but any command sent directly to the virtual address of 10.10.10.2 simply goes unanswered.

[0006] This is a serious limitation for a network manager or architect who oftentimes needs to verify that the hardware on the computer network is working properly. Typically, a network troubleshooter would send ping or traceroute commands to test and verify that the network hardware is correctly operating. However, in present virtual routing systems, receiving useful information in response to such commands sent to a virtual router is presently not possible.

[0007] The present invention addresses these and other problems associated with the prior art.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] **FIG. 1** is a block diagram of a network device according to the prior art.

[0009] **FIG. 2** is a sample static route table for a virtual router network according to an embodiment of the invention.

[0010] **FIG. 3** is an example flow diagram illustrating processes that can be used to set up a VRRP network according to embodiments of the invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0011] Embodiments of the invention create an association between one or more virtual addresses and a loopback interface of a networked device. Once the virtual addresses are assigned to the loopback interface, any commands addressed to one of the assigned virtual addresses proceeds through the loopback of the network device and is acted upon by the protocol stack. Thus, commands that produce valuable feedback information, like "ping" and "traceroute" can be sent directly to the virtual router address, and acted on by the physical machine currently serving as the virtual router.

[0012] Embodiments of the invention use the virtual IP address as a modified "static route". Static routes can be created regardless of which subnet the virtual IP address belong to. However, a router having a standard static route doesn't "own" the route, and hence cannot accept traffic destined for that static address.

[0013] Instead of pointing the static route to an interface or IP address of a neighboring router (as static routes are created), the static route is pointed to the internal loopback address 127.x.x.x. IP addresses beginning with 127 are reserved on all machines to be used for testing the machine itself. Frequently the loopback address is given as 127.0.0.1. This address tells the router that any IP packets destined for the virtual IP address should be sent to "itself", and passed up into the IP stack of the machine to be handled by an application running on the machine itself. For instance, if packets making up a ping command arrive destined for the virtual router address, the physical router presently serving as the virtual router will send the packets to its own loopback address. Then, the IP stack of the router will pass them up to the ping application running on the router itself, rather

than forwarding them out of the router. When the ping application sees the ping command, it responds back as it normally would, by placing its address in the ping payload and sending notice back to the source that sent the ping command. Embodiments of the invention could operate by either sending the virtual address back as the payload data, or the physical address of the router acting as the virtual router could be placed in the payload instead.

[0014] The static route can be pointed to the internal loopback address when the router is initially being set up to be a virtual router, or, the static route can be modified dynamically when a slave router becomes the master router after the master router fails, for instance.

[0015] **FIG. 2** shows a sample static routing table **10** for a router configured as a virtual router according to an embodiment of the invention. Illustrated in the left column is the destination IP address. Illustrated in the right column is the address of the interface within the router, or the IP address of the neighboring router to which data sent to the destination IP address is sent. Note that packets sent to the IP address of the virtual router, 10.10.10.2, are statically referenced to the loopback address of 127.0.0.1.

[0016] Assigning a Host Address to Machine

[0017] Embodiments of the invention, by routing packets destined for a particular address to the machine itself by using the loopback address, act similar to a "host" using host addressing. Specifically indicating that IP commands sent to the virtual address should be acted on by the physical router presently acting as the virtual router is similar to indicating that a particular machine is a host. Commands or programs can be performed on the router from another machine, such as telnet. Or, the router ID could be used for protocols, such as OSPF (Open Shortest Past First), ISIS (Intermediate System to Intermediate System) or BGP (Border Gateway Protocol).

[0018] In routers according to the prior art, OSPF picks up the IP addresses assigned to specific interfaces within the router as a router ID. Using embodiments of the invention, an IP address (the virtual IP address) is assigned to the router itself, rather than a specific interface. One advantage to this procedure is that, if any specific interface should fail, the OSPF process need not be reset to operate correctly. In other words, because an IP address can be assigned to a router, and is not limited to being assigned to a single interface, as long as the router itself is operational, then the IP address will always be available to the OSPF process.

[0019] Setting up the VRRP Network

[0020] **FIG. 3** illustrates processes **100** used in establishing and configuring a VRRP network. In a first process **110**, a virtual router is defined by first creating an identification number for the virtual router. Typically, this is performed at the interface level, i.e. for a particular interface in the router. A sample command to assign the identification number to the particular interface is "vrrp-group vrid", where vrid is the identification number for the virtual router. Conversely, to delete an identification number from the interface, the command "no vrrp-group vrid" is used.

[0021] Virtual routers contain virtual IP addresses configured for that VRRP group, in addition to other configuration information. In some embodiments of the invention, a VRRP group is unable to transmit VRRP packets until a virtual IP addresses is assigned to the virtual group id. To activate a VRRP group on an interface, which will cause the VRRP group to start transmitting IP packets, an IP number is assigned to the IP group, as illustrated in process **120**. A sample command is "virtual-address ip-address1[ip-address2] . . . [ip-address 12]", where ip-address is the IP address of the virtual routers, and does not include the IP address mask. In some embodiments of the invention, as illustrated above, up to 12 virtual ip-addresses can be configured for a VRRP group.

[0022] In some cases it is beneficial to set priorities for the VRRP group. One benefit to setting priorities is to indicate which physical router interface takes the master role for the virtual router. In other words, the router that is configured to be the master router will take over primary responsibility for routing packets addressed by the host to the virtual router. If, for some reason, the master router should fail, one or more slave routers take responsibility for the virtual router. Setting the priorities allows the network architect to determine in which order the routers in the VRRP will assume the virtual routing role. Setting the priority is illustrated as process **130**, and, in some embodiments, can be accomplished with the command "priority priority" where the priority number is a number between 1 and 255. If no priority is specifically set for a router, its default priority is 100.

[0023] In some cases, it is desirable to force the master router to automatically defer to another router that comes on the network having a higher priority than the current master router. A command to enable such preemption is illustrated as process **140** and can be accomplished with the command "preempt". To turn off preemption, a command "no preempt" is entered. In some embodiments, the preemption command is turned on by default.

[0024] Setting simple authentication of VRRP packets ensures that only trusted routers participate in VRRP processes. When authentication is enabled, a password is included in the VRRP packets and a receiving router uses the password to verify the transmission. Setting authentication is illustrated as process **150**, and, in some embodiments, can be accomplished with the command "authentical-type simple [encryption type] password", where encryption type identifies how the password is encrypted, and password is the password itself.

[0025] In the VRRP network, the currently operating master router advertises to all of the members of the VRRP group that it is running as the master router. If the VRRP group does not receive an advertisement, then an election process begins and the backup virtual router having the highest priority (after the disabled master) transitions to the master. In some embodiments, it may be beneficial to change the advertisement time, as illustrated as process **160**. In some embodiments, the command can be implemented as "advertise-interval seconds" where seconds indicates the number of seconds between master router advertisements. The default can be set to be one, for example, and to have an overall range of between 1-255.

[0026] In some embodiments it may be beneficial to monitor an interface used by a virtual group. The virtual group sends VRRP packets out from the interface and, if the interface is disabled or shutdown, the VRRP packets are not transmitted. With tracking enabled, embodiments of the

invention can lower the priority of the VRRP group(s) on that interface if the interface is disabled or goes down. The lowered priority of the VRRP group may trigger an election for a new master router. To track an interface, as illustrated in process **170**, a command can be given such as "track interface [priority-cost cost]", where interface indicates which interface is in the particular VRRP, and priority-cost is a value to be subtracted from the group priority should the interface go down. The priority cost can have a value of, for instance, 1-255, with a default of 10.

[0027] Described herein is only a sample system which may look much different than an actual system in implementation. The system described above can use dedicated processor systems, micro controllers, programmable logic devices, or microprocessors that perform some or all of the processing and routing functions. Some of the operations described above may be implemented in software and other operations may be implemented in hardware.

[0028] Of course there can be many or few routers in the VRRP network and many or few interfaces within each router. Additionally, not all of the processes shown in **FIG. 3** need to be set up in all cases. In some embodiments, default values can be assigned. As always, implementation is left to the system designer, and many of the specific details may be best determined empirically.

[0029] For the sake of convenience, the operations are described as various interconnected functional blocks or distinct modules. This is not necessary, however, and there may be cases where these functional blocks or modules are equivalently aggregated into a single logic device, program or operation with unclear boundaries. In any event, the functional blocks and software modules or described features can be implemented by themselves, or in combination with other operations in either hardware or software.

[0030] Having described and illustrated the principles of the invention in a preferred embodiment thereof, it should be apparent that the invention may be modified in arrangement and detail without departing from such principles. Claim is made to all modifications and variation coming within the spirit and scope of the following claims.

What is claimed is:

1. A virtual router network, comprising:

a host coupled to the network;

a first router coupled to the host and having a first routing table with an entry relating a virtual address to an address for the first router; and

a second router coupled to the host and having a second routing table with an entry relating the virtual address to an address for the second router.

2. The virtual router network of claim 1 wherein:

the address for the first router is a loopback address for the first router; and

the address for the second router is a loopback address for the second router.

3. The virtual router network of claim 1 wherein the router network is a VRRP network.

4. The virtual router network of claim 1 wherein the loopback address for the first router is 127.x.x.x, where x indicates any integer between 0 and 255.

5. The virtual router network of claim 1, wherein, at any given time, only one routers is structured to respond to commands addressed to the virtual address.

6. The virtual router network of claim 1, wherein:

the first router is a master router; and

the second router is a slave router.

7. The virtual router network of claim 6 wherein one of the routers serves as an OSPF router.

8. A method for assigning a virtual router address to a network of computer devices, the method comprising:

assigning a virtual router address to a loopback address of a first computer device; and

assigning the virtual router address to a loopback address of a second computer device, the second computer device coupled to the first computer device.

9. The method of claim 8, further comprising assigning more than one virtual router addresses to the loopback address of the first computer device.

10. The method of claim 9 wherein assigning more than one virtual router addresses comprises assigning up to twelve virtual router addresses.

11. The method of claim 8, further comprising:

assigning a priority to the first computer device; and

assigning a priority to the second computer device, the priority assigned to the first computer device higher than the priority assigned to the second computer device.

12. The method of claim 11 wherein the first computer device acts as a master virtual router, the method further comprising setting an indicator on the first computer device to pass the role of master virtual router to any router that couples to the network of computer devices that has a priority number higher than the priority assigned to the first computer device.

13. A virtual routing network, comprising:

a static routing table in a network communication device coupled to the virtual routing network; and

an entry in the static routing table assigning a virtual routing address to a loopback address of the network communication device.

14. The virtual routing network of claim 13 wherein the virtual routing address is a virtual IP address.

15. The virtual routing network of claim 13, further comprising:

an IP protocol stack on the network communication device.

16. The virtual routing network of claim 15, further comprising:

one or more applications operating on the network communication device and structured to operate on the IP protocol stack.

17. The virtual routing network of claim 13, further comprising:

a second network communication device coupled to the network communication device, the second network communication device including a static routing table

having an entry assigning the virtual routing address to a loopback address of the second communication device.

18. The virtual routing network of claim 13 wherein the network is a VRRP.

19. The virtual routing network of claim 13 wherein the network communication device is structured to operate on commands destined for the virtual routing address.

20. The virtual routing network of claim 17 wherein at any given time, only one of the network communication device and the second network communication device is structured to operate on commands destined for the virtual routing address.

* * * * *

US 20020023210A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0023210 A1

Tuomenoksa et al. (43) Pub. Date: Feb. 21, 2002

(54) METHOD AND SYSTEM FOR MANAGING AND CONFIGURING VIRTUAL PRIVATE NETWORKS

(76) Inventors: **Mark Tuomenoksa**, Winchester, MA (US); **Samuel Bendinelli**, Princeton, NJ (US); **Jerold Francus**, Far Hills, NJ (US); **Jonathan Harwood**, Rumson, NJ (US); **Michael Herrick**, Colts Neck, NJ (US); **John Keane**, Metuchen, NJ (US); **Christopher Macey**, Red Bank, NJ (US); **Brion Shimamoto**, Riverside, CT (US)

Correspondence Address:
Finnegan, Henderson, Farabow
Garrett & Dunner, L.L.P.
1300 I Street, N.W.
Washington, DC 20005-3315 (US)

(21) Appl. No.: **09/814,178**

(22) Filed: **Mar. 22, 2001**

Related U.S. Application Data

(63) Non-provisional of provisional application No. 60/196,297, filed on Apr. 12, 2000.

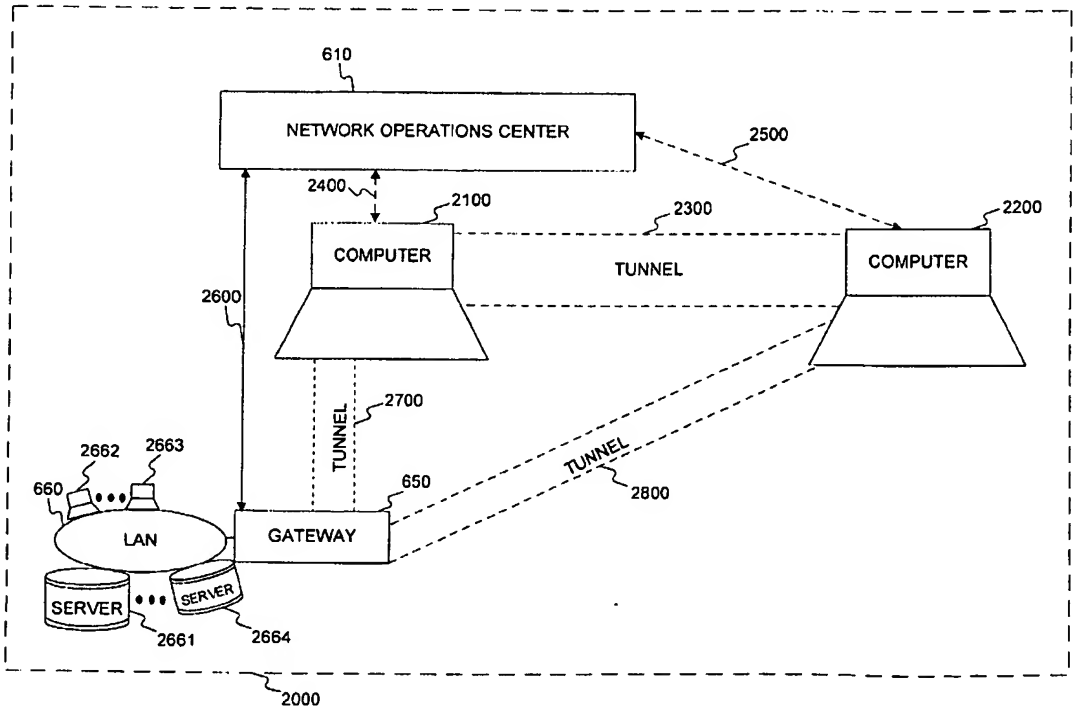
Publication Classification

(51) **Int. Cl.**[7] **H04L 9/00**

(52) **U.S. Cl.** **713/161**; 713/153; 713/166; 713/168; 713/170

(57) **ABSTRACT**

Methods and systems are provided for enabling a network between a first and a second processor using at least one additional processor separate from the first and second processors. In one embodiment, the at least one additional processor receives information indicating a consent on behalf of the first processor to enabling a tunnel between the first processor and the second processor and receives information indicating a consent on behalf of the second processor to enabling a tunnel between the second processor and the first processor. The at least one additional processor determines a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the network. The at least one additional processor provides to each of the first and second processors the first and second virtual addresses to enable one or more tunnels between the first and the second processors.



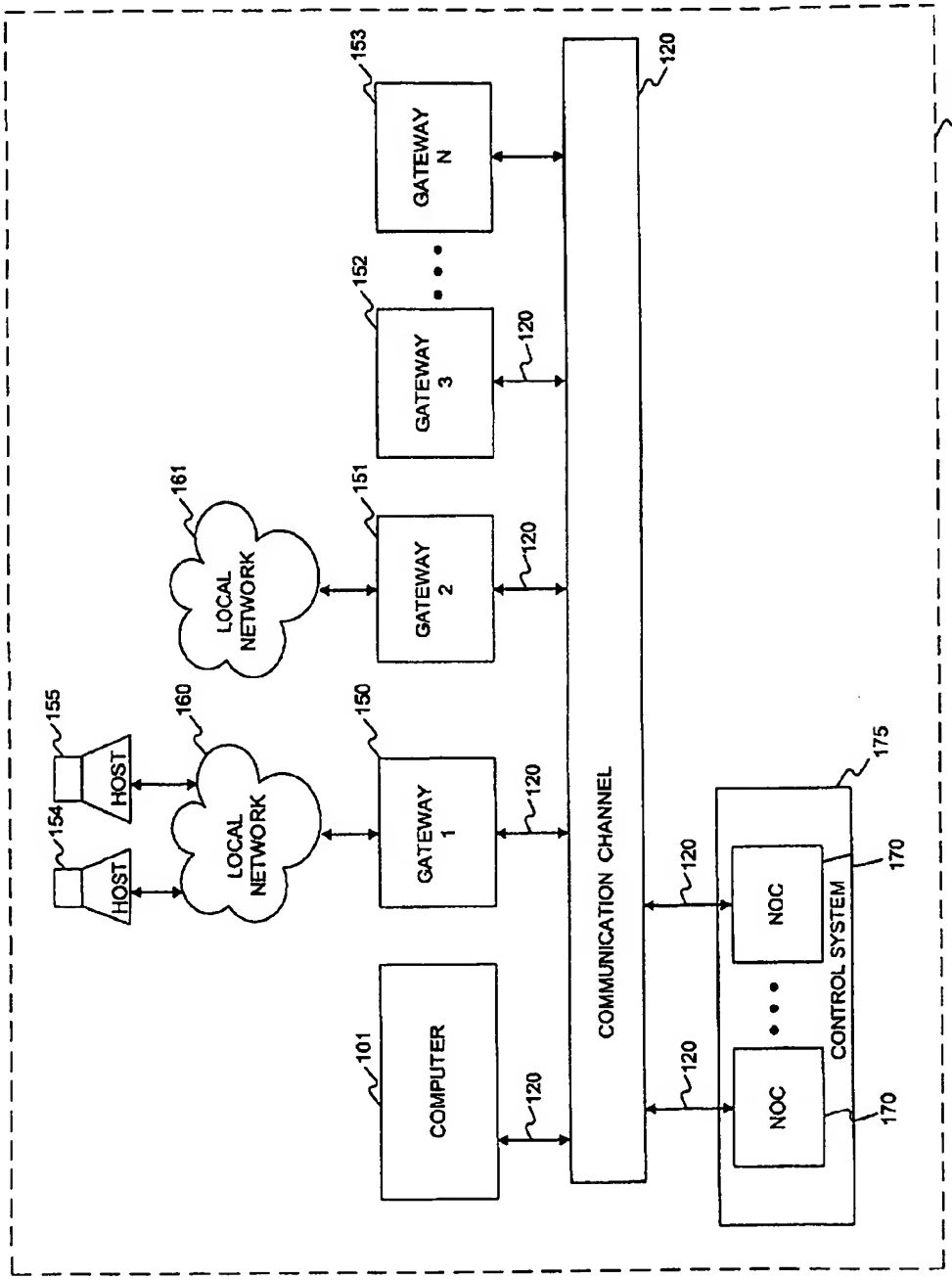

100
COMMUNICATION CHANNEL
120
COMPUTER
101
GATEWAY 1
150
GATEWAY 2
151
GATEWAY 3
152
GATEWAY N
153
LOCAL NETWORK
160
LOCAL NETWORK
161
HOST
154
HOST
155
CONTROL SYSTEM
175
NOC
170
NOC
170

FIG. 1

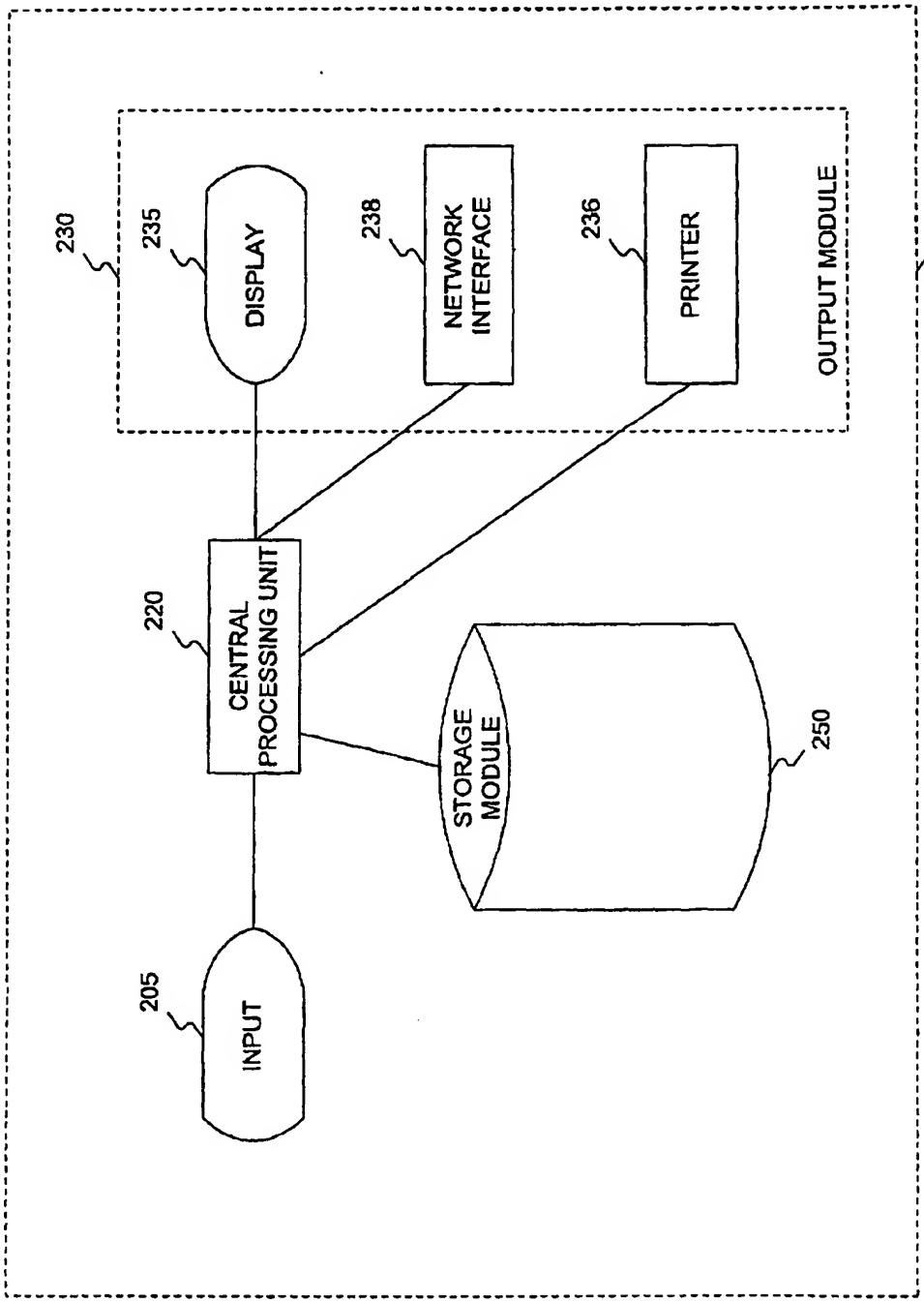

230
235
DISPLAY
238
NETWORK INTERFACE
236
PRINTER
OUTPUT MODULE
200
220
CENTRAL PROCESSING UNIT
STORAGE MODULE
250
205
INPUT

FIG. 2

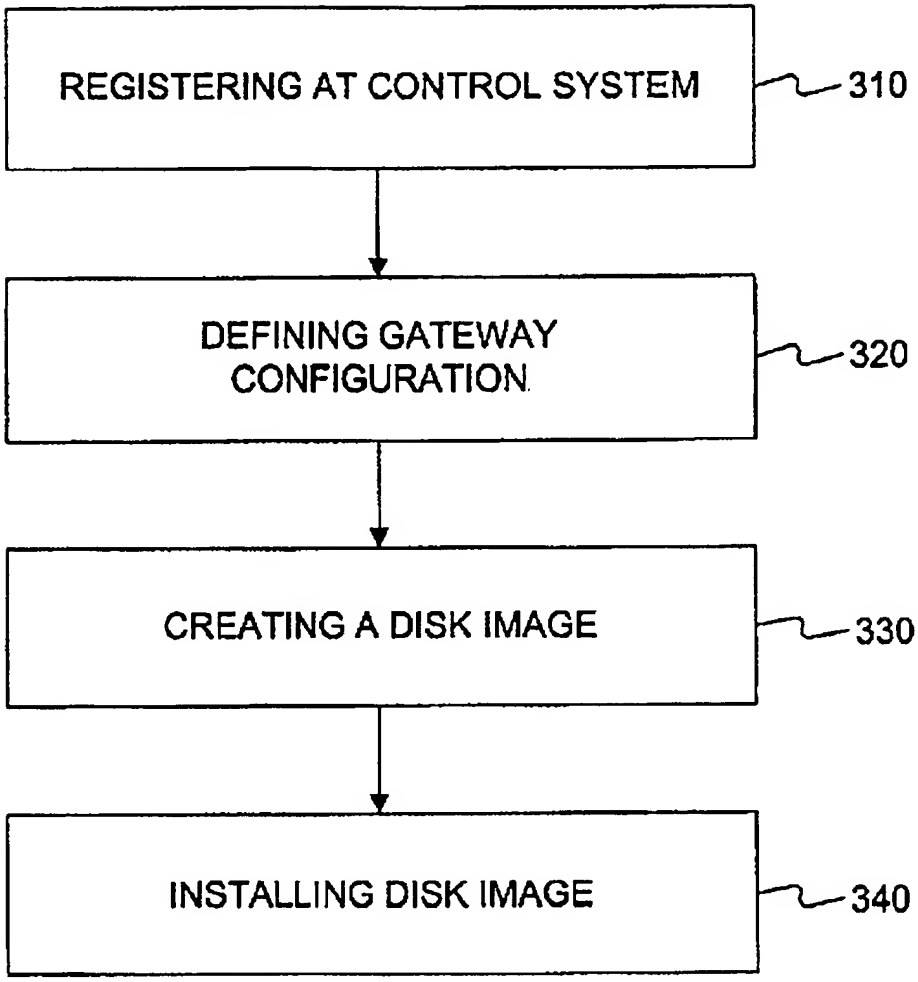

REGISTERING AT CONTROL SYSTEM
310
DEFINING GATEWAY CONFIGURATION
320
CREATING A DISK IMAGE
330
INSTALLING DISK IMAGE
340

FIG. 3

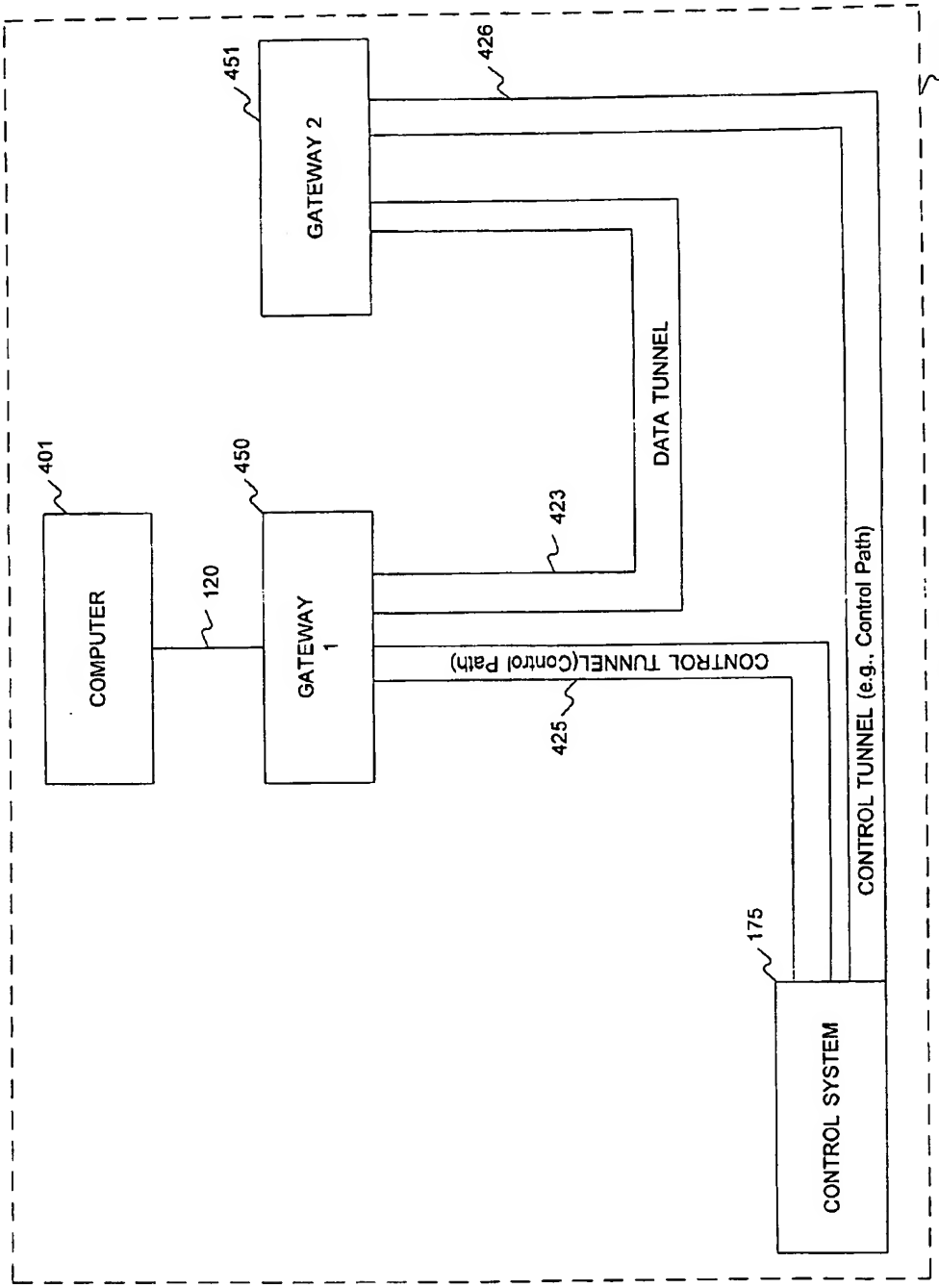

400
451
GATEWAY 2
426
DATA TUNNEL
401
450
423
COMPUTER
120
GATEWAY 1
CONTROL TUNNEL(Control Path)
425
CONTROL TUNNEL (e.g., Control Path)
175
CONTROL SYSTEM

FIG. 4

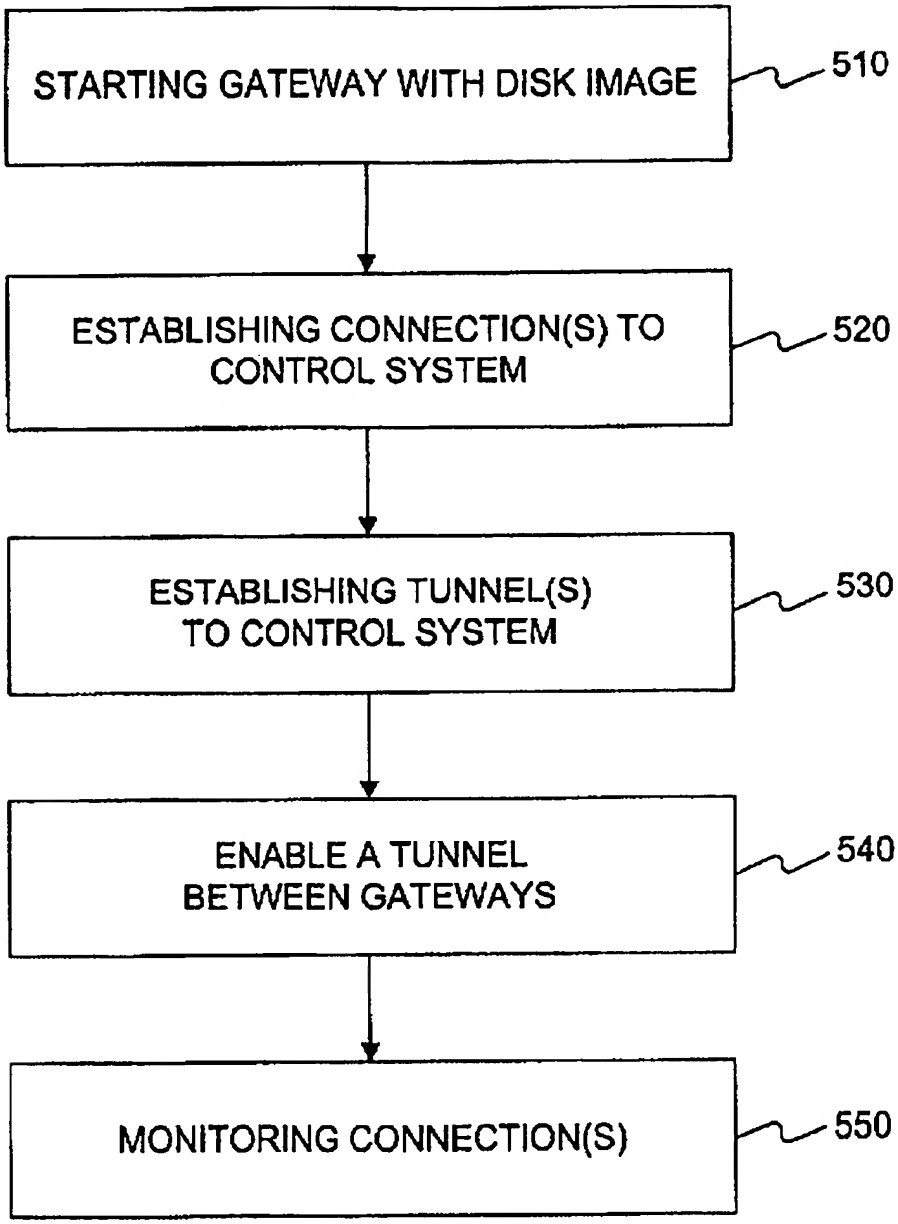

STARTING GATEWAY WITH DISK IMAGE
510
ESTABLISHING CONNECTION(S) TO CONTROL SYSTEM
520
ESTABLISHING TUNNEL(S) TO CONTROL SYSTEM
530
ENABLE A TUNNEL BETWEEN GATEWAYS
540
MONITORING CONNECTION(S)
550

FIG. 5

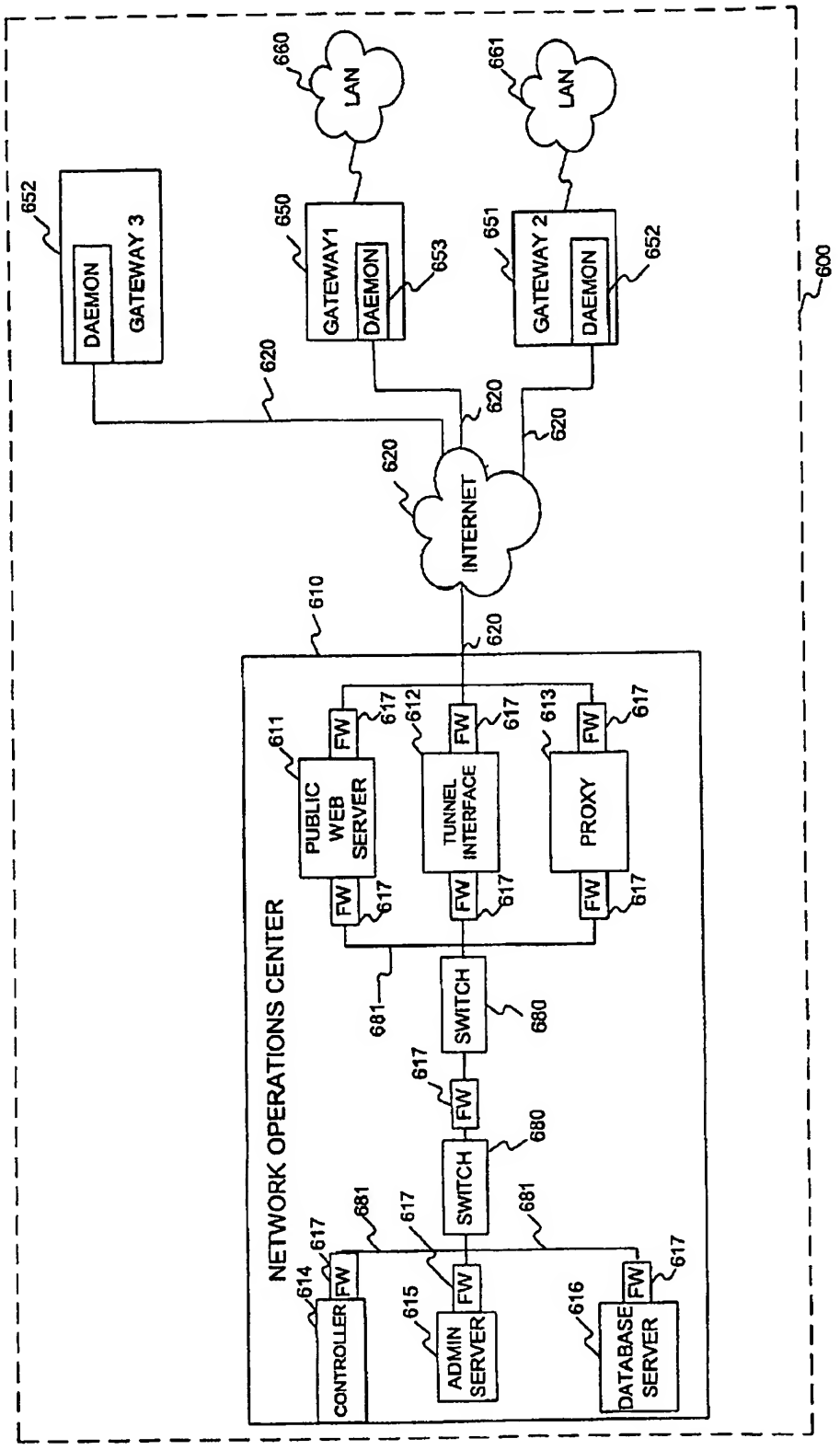

600
660
LAN
661
LAN
652
DAEMON
GATEWAY 3
650
GATEWAY1
DAEMON
653
651
GATEWAY 2
DAEMON
652
620
620
620
620
INTERNET
610
620
NETWORK OPERATIONS CENTER
611
PUBLIC WEB SERVER
612
TUNNEL INTERFACE
613
PROXY
FW
617
681
SWITCH
680
617
FW
680
SWITCH
681
617
681
617
614
CONTROLLER
615
ADMIN SERVER
616
DATABASE SERVER

FIG. 6A

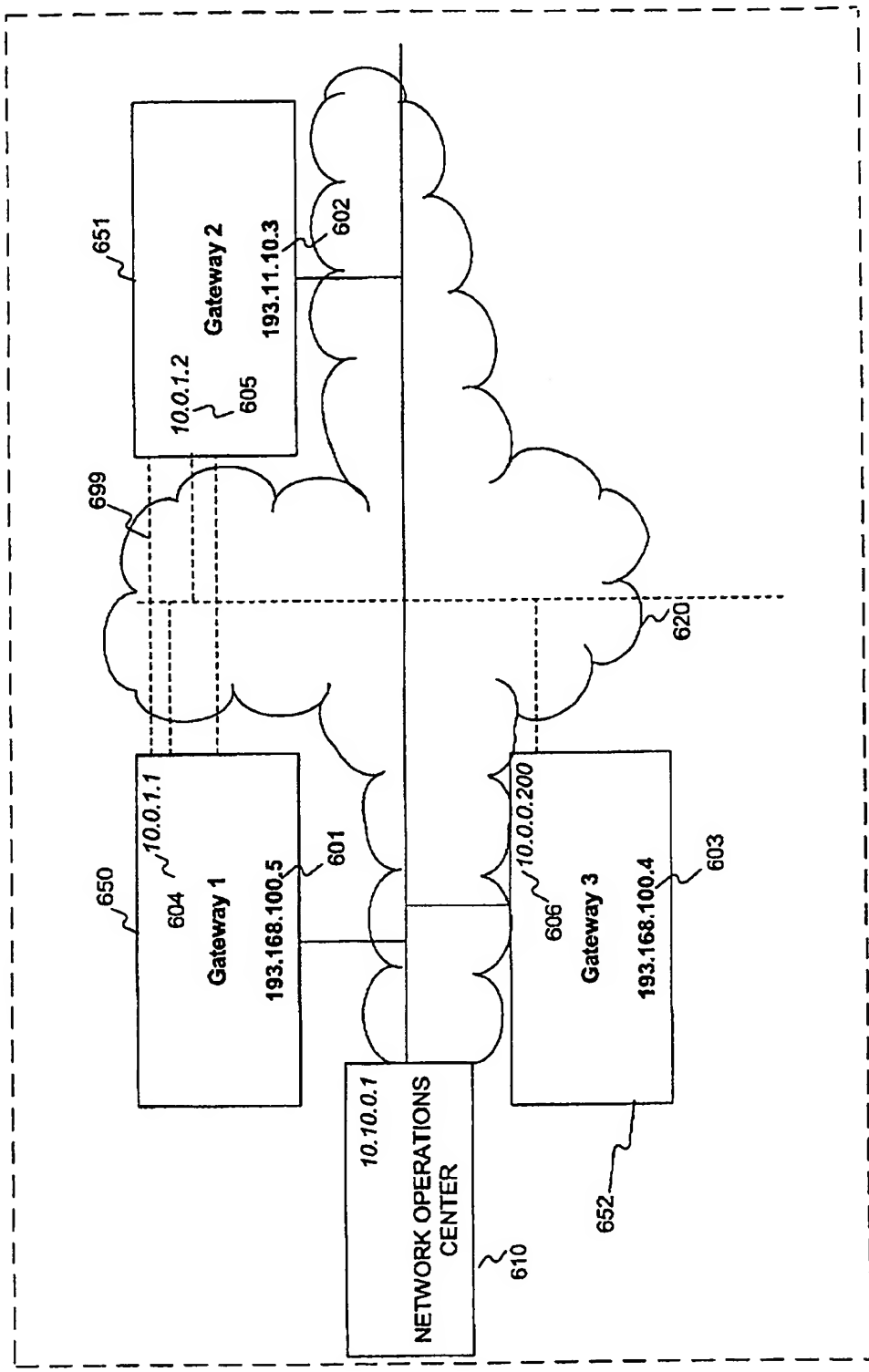

651
Gateway 2
193.11.10.3
602
10.0.1.2
605
699
620
650
10.0.1.1
604
Gateway 1
193.168.100.5
601
10.0.0.200
606
Gateway 3
193.168.100.4
603
10.10.0.1
NETWORK OPERATIONS CENTER
610
652
600

FIG. 6B

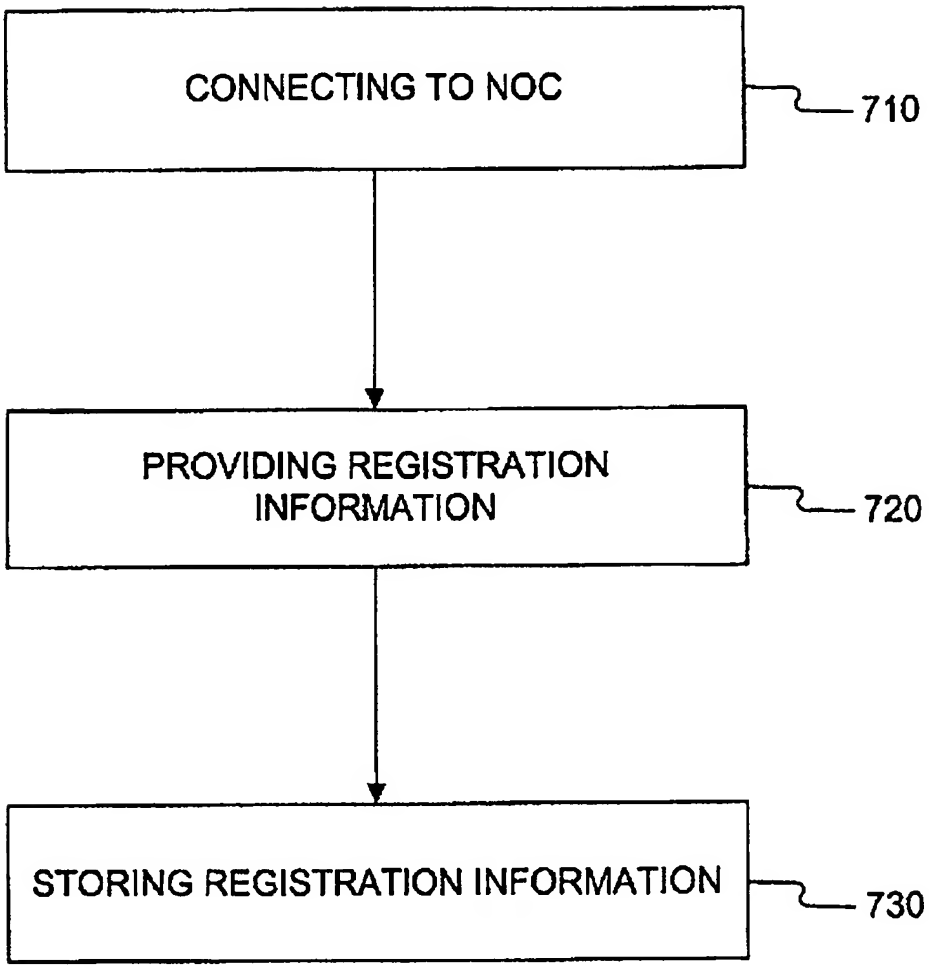

CONNECTING TO NOC
710
PROVIDING REGISTRATION INFORMATION
720
STORING REGISTRATION INFORMATION
730

FIG. 7

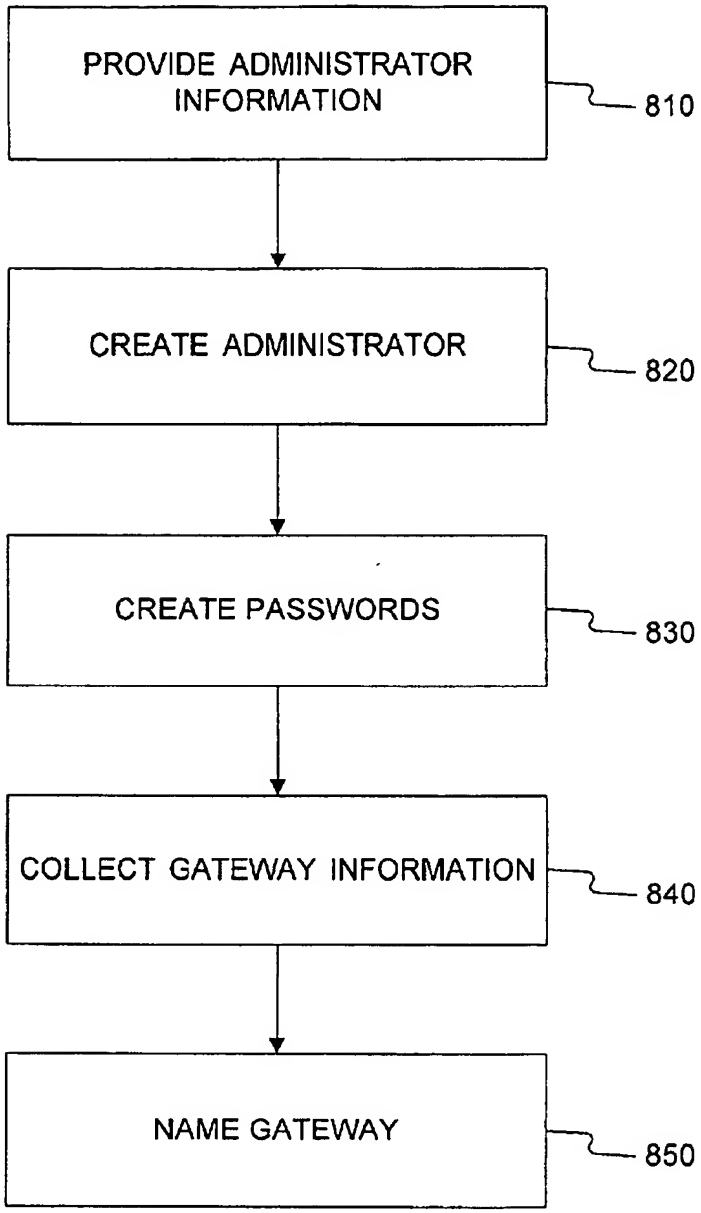

PROVIDE ADMINISTRATOR INFORMATION
810
CREATE ADMINISTRATOR
820
CREATE PASSWORDS
830
COLLECT GATEWAY INFORMATION
840
NAME GATEWAY
850

FIG. 8

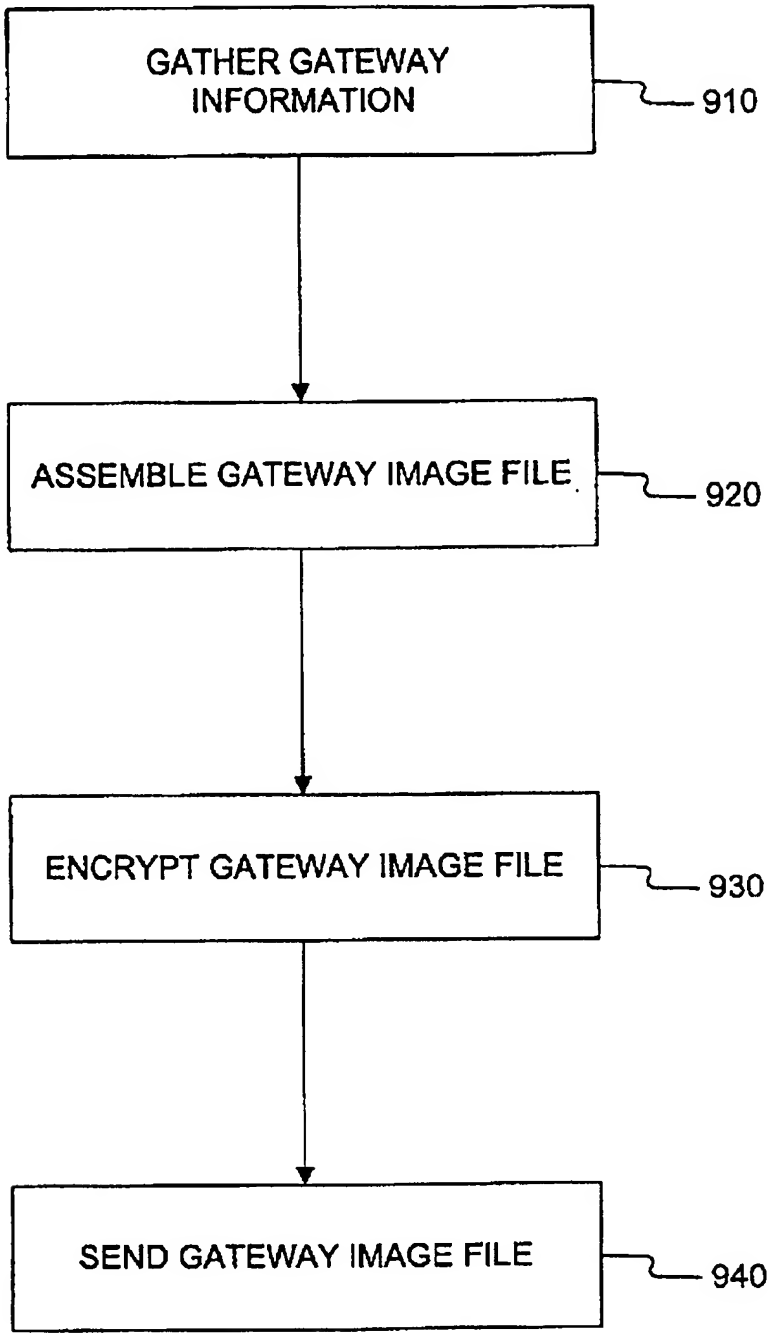

GATHER GATEWAY INFORMATION
910
ASSEMBLE GATEWAY IMAGE FILE
920
ENCRYPT GATEWAY IMAGE FILE
930
SEND GATEWAY IMAGE FILE
940

FIG. 9

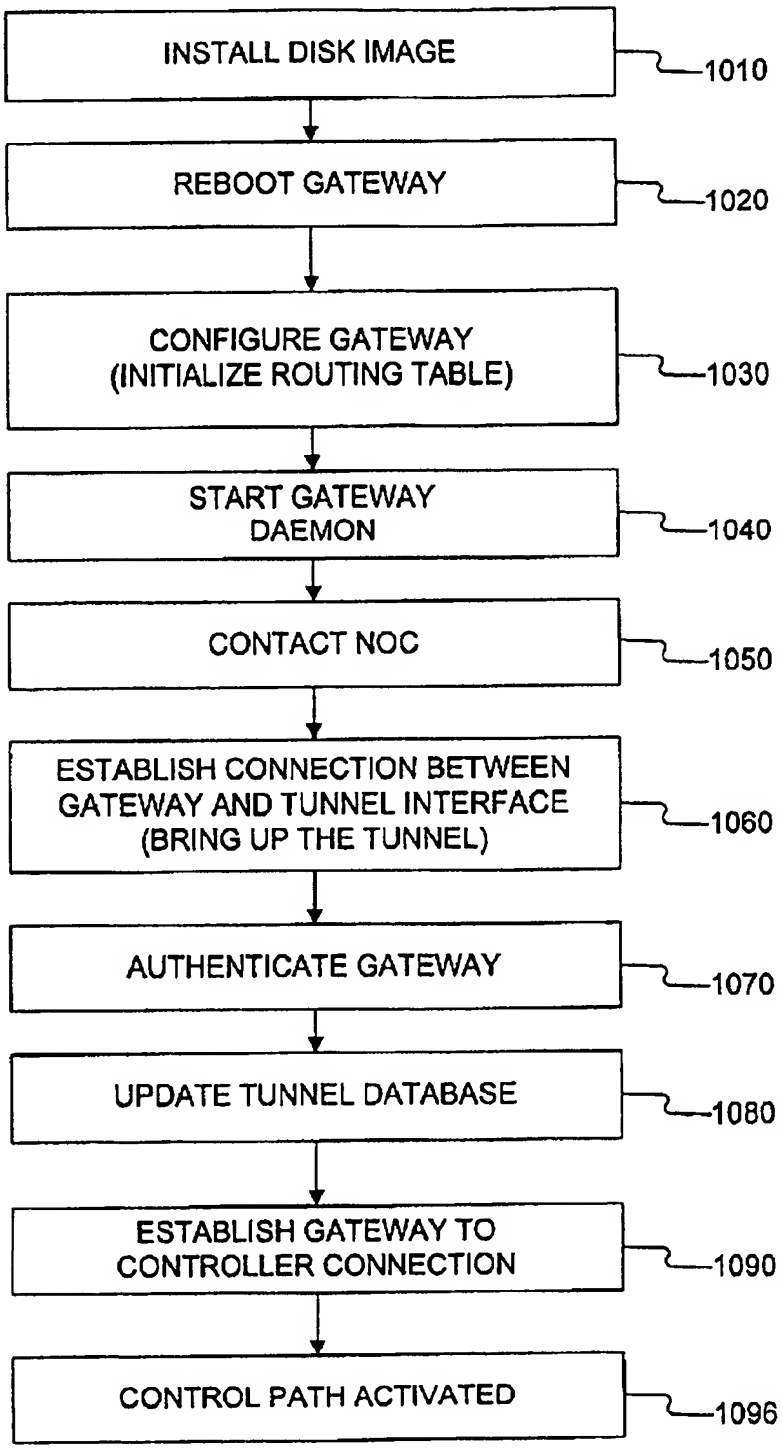

INSTALL DISK IMAGE
1010
REBOOT GATEWAY
1020
CONFIGURE GATEWAY (INITIALIZE ROUTING TABLE)
1030
START GATEWAY DAEMON
1040
CONTACT NOC
1050
ESTABLISH CONNECTION BETWEEN GATEWAY AND TUNNEL INTERFACE (BRING UP THE TUNNEL)
1060
AUTHENTICATE GATEWAY
1070
UPDATE TUNNEL DATABASE
1080
ESTABLISH GATEWAY TO CONTROLLER CONNECTION
1090
CONTROL PATH ACTIVATED
1096

FIG. 10

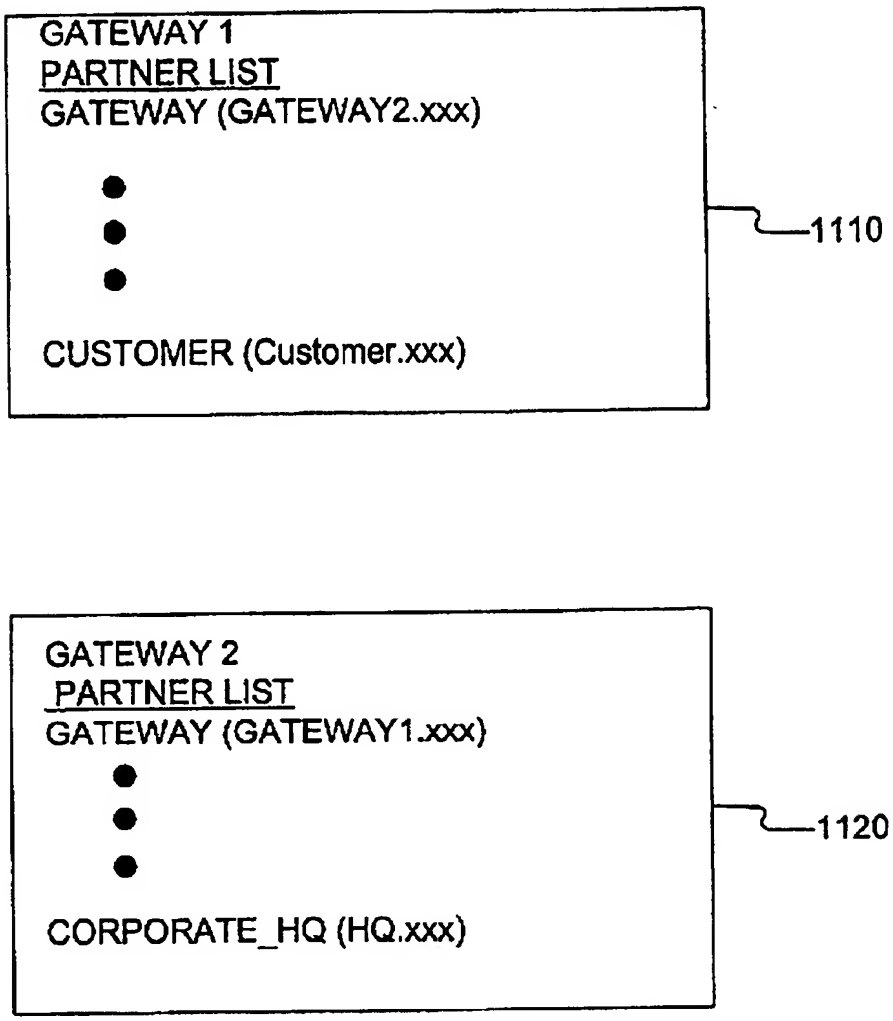

GATEWAY 1
PARTNER LIST
GATEWAY (GATEWAY2.xxx)
CUSTOMER (Customer.xxx)
1110
GATEWAY 2
PARTNER LIST
GATEWAY (GATEWAY1.xxx)
CORPORATE_HQ (HQ.xxx)
1120

FIG. 11

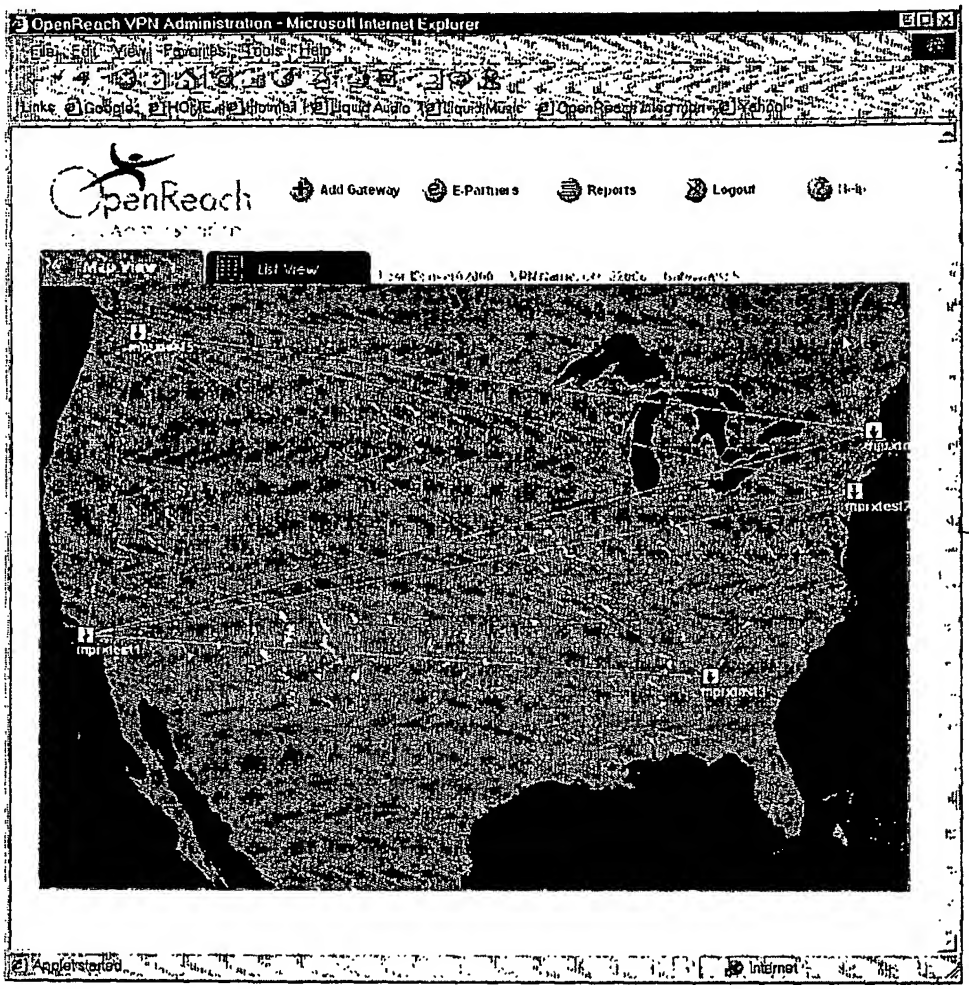

OpenReach VPN Administration - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
OpenReach
Add Gateway
E-Partners
Reports
Logout
Map View
List View
1250

FIG. 12

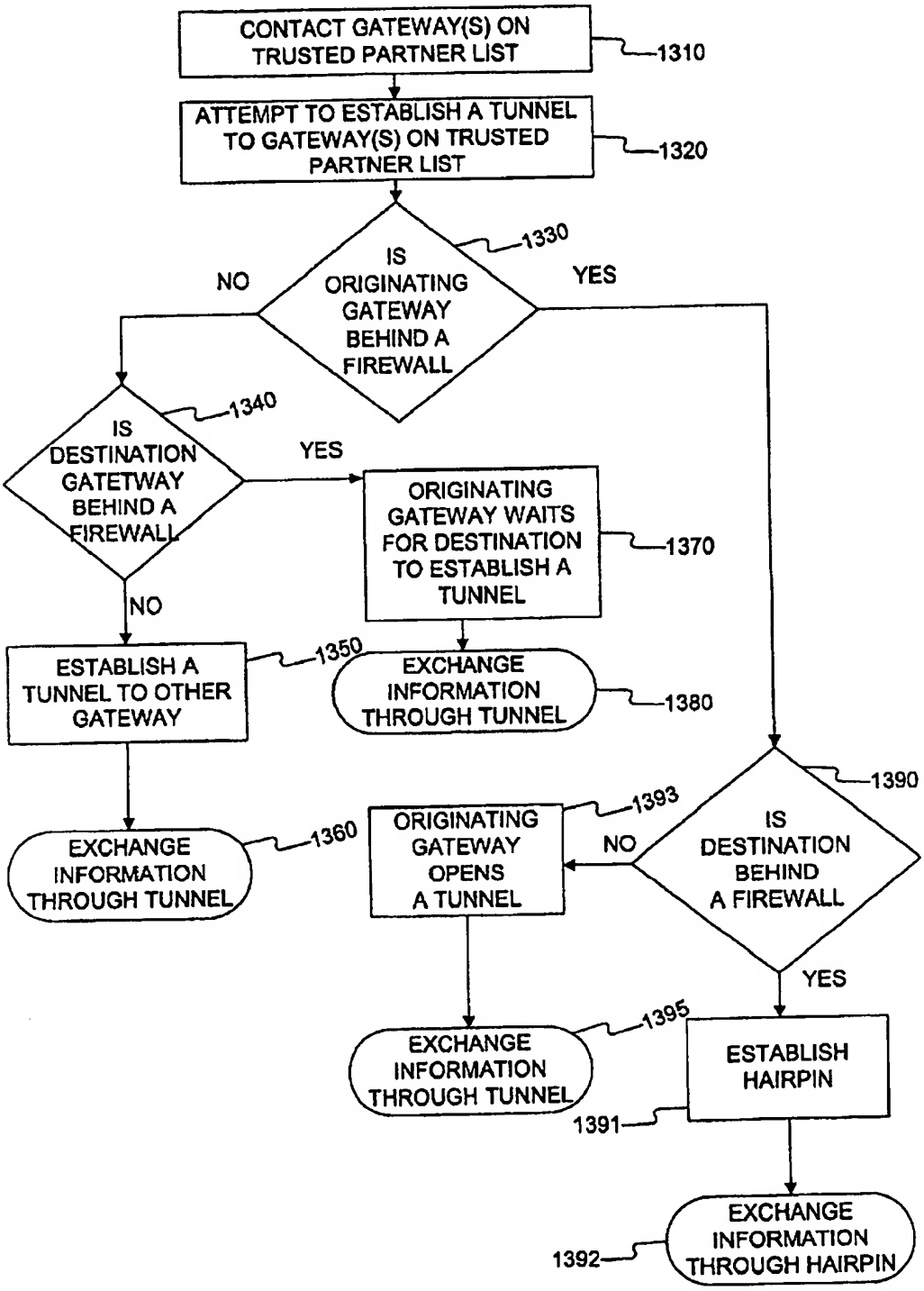

CONTACT GATEWAY(S) ON TRUSTED PARTNER LIST
1310
ATTEMPT TO ESTABLISH A TUNNEL TO GATEWAY(S) ON TRUSTED PARTNER LIST
1320
IS ORIGINATING GATEWAY BEHIND A FIREWALL
1330
NO
YES
IS DESTINATION GATETWAY BEHIND A FIREWALL
1340
YES
NO
ORIGINATING GATEWAY WAITS FOR DESTINATION TO ESTABLISH A TUNNEL
1370
ESTABLISH A TUNNEL TO OTHER GATEWAY
1350
EXCHANGE INFORMATION THROUGH TUNNEL
1380
EXCHANGE INFORMATION THROUGH TUNNEL
1360
ORIGINATING GATEWAY OPENS A TUNNEL
1393
IS DESTINATION BEHIND A FIREWALL
1390
NO
YES
EXCHANGE INFORMATION THROUGH TUNNEL
1395
ESTABLISH HAIRPIN
1391
EXCHANGE INFORMATION THROUGH HAIRPIN
1392

FIG. 13

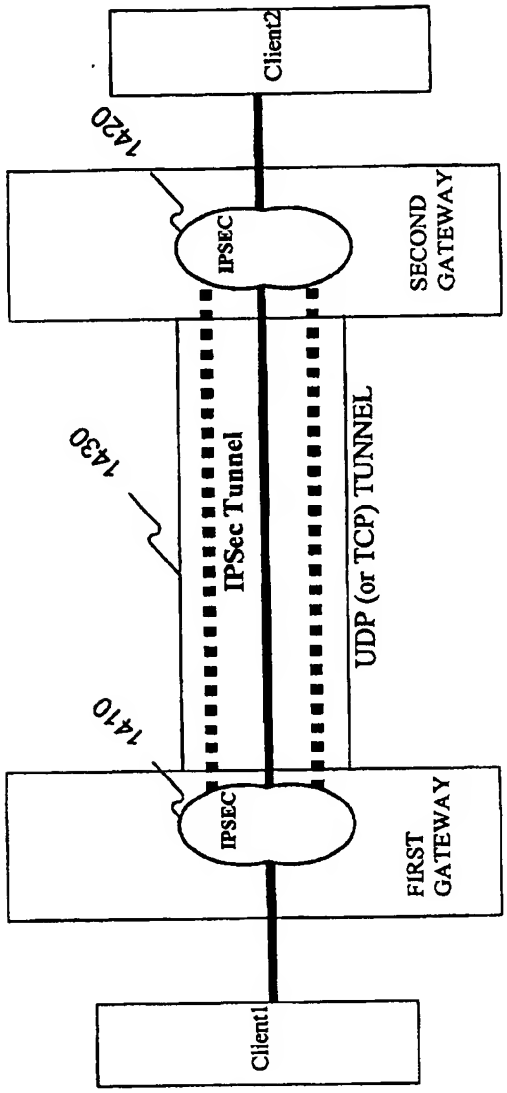

Client2
1420
IPSEC
SECOND GATEWAY
1430
IPSec Tunnel
UDP (or TCP) TUNNEL
1410
IPSEC
FIRST GATEWAY
Client1

FIG. 14

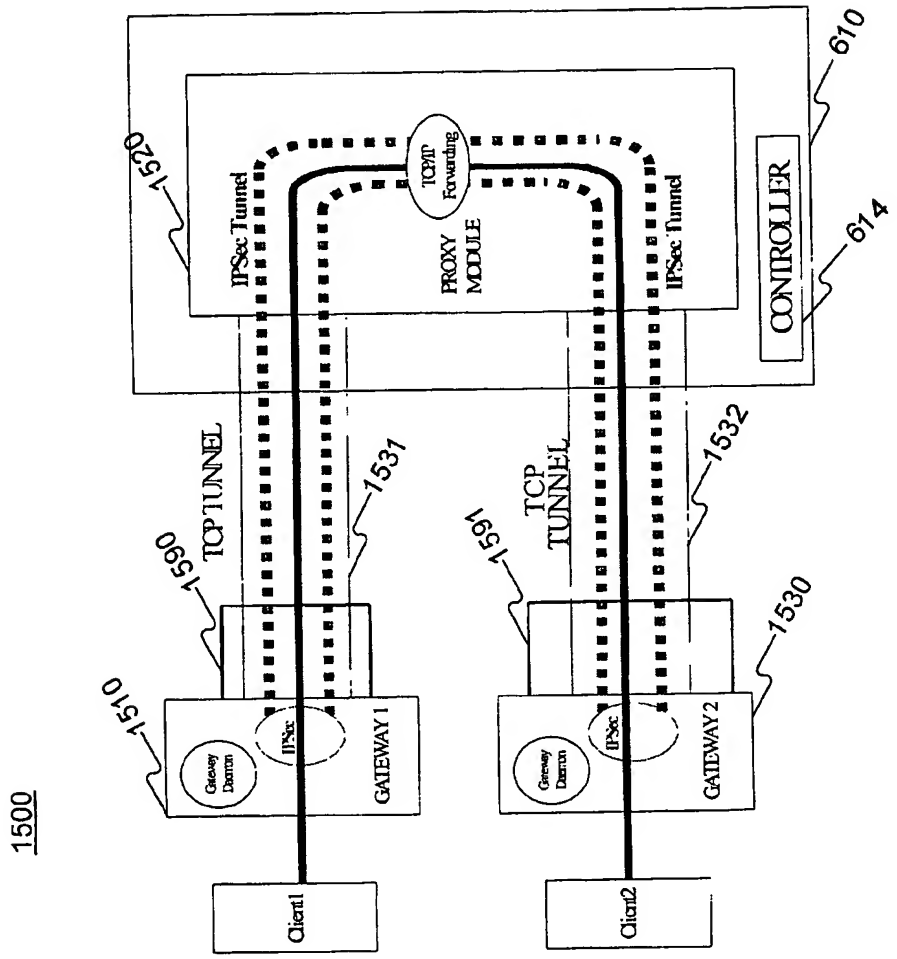

1500
610
1520
IPSec Tunnel
TCP/IP Forwarding
PROXY MODULE
IPSec Tunnel
CONTROLLER
614
TCP TUNNEL
1537
1532
1590
1591
1530
1510
Gateway Daemon
IPSec
GATEWAY 1
GATEWAY 2
Client1
Client2

FIG. 15

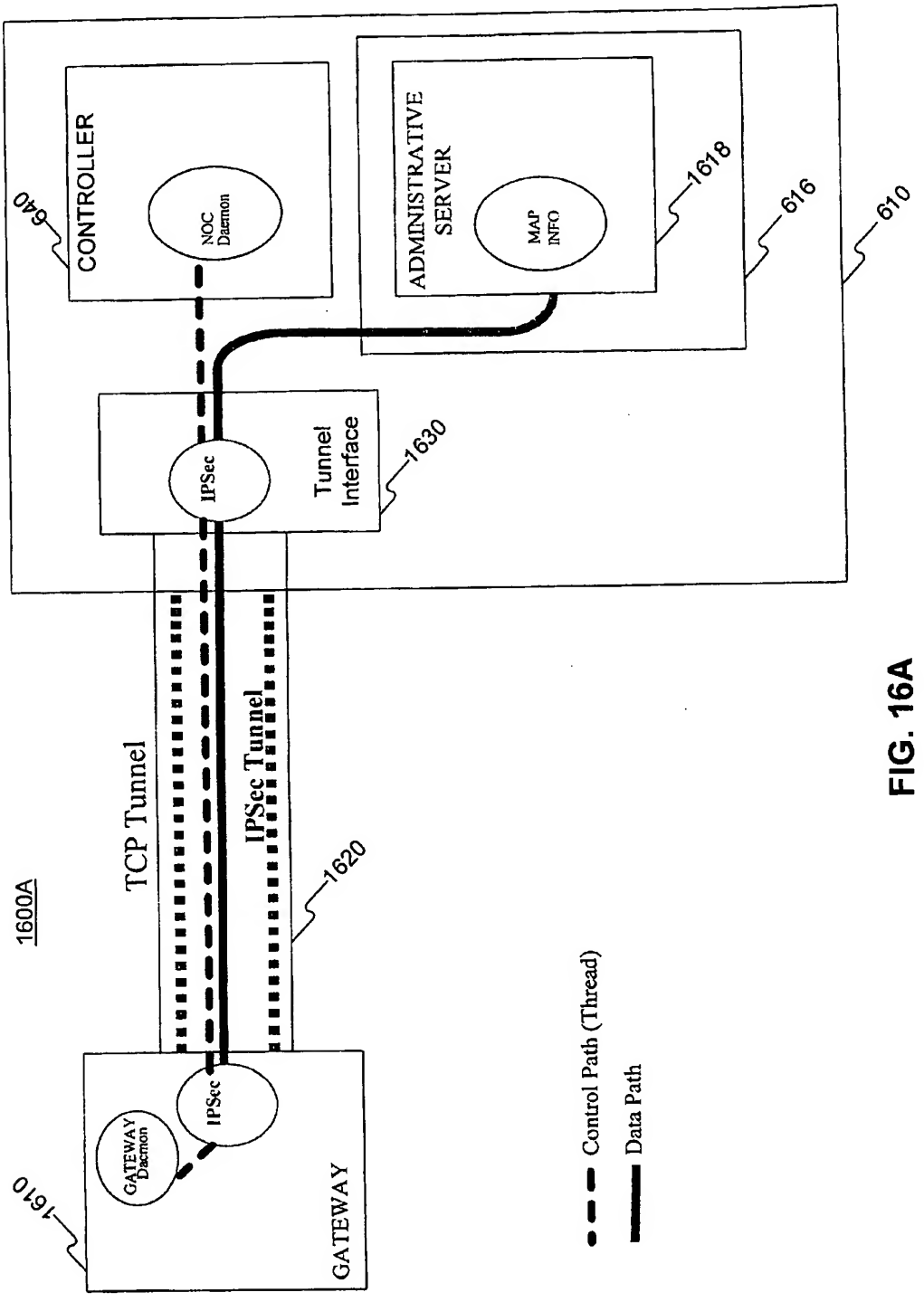

1600A
640
CONTROLLER
NOC Daemon
ADMINISTRATIVE SERVER
MAP INFO
1618
616
610
IPSec
Tunnel Interface
1630
TCP Tunnel
IPSec Tunnel
1620
GATEWAY Daemon
IPSec
GATEWAY
1610
Control Path (Thread)
Data Path

FIG. 16A

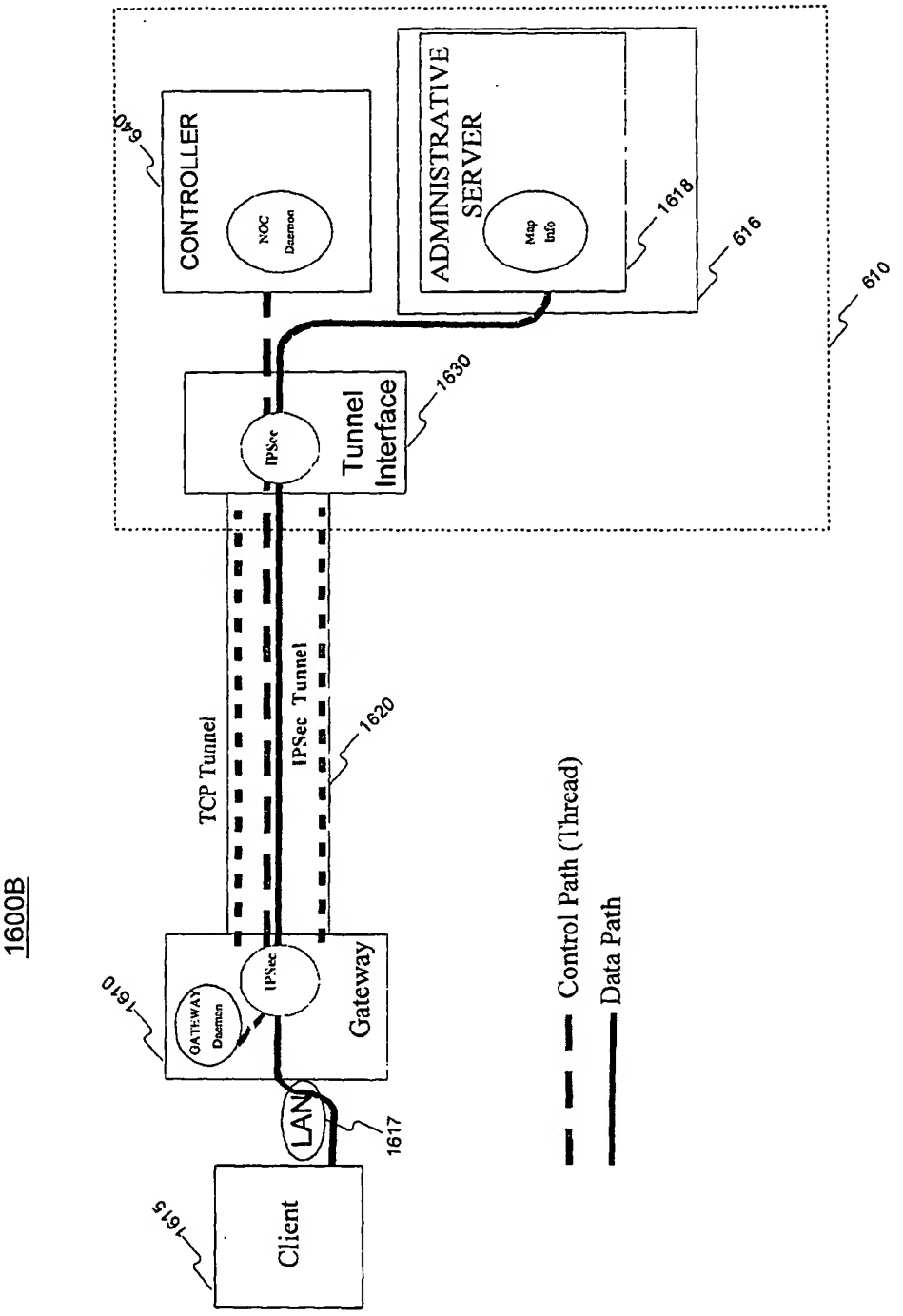

1600B
CONTROLLER
NOC Daemon
640
ADMINISTRATIVE SERVER
Map Info
1618
616
610
IPSec
Tunnel Interface
1630
TCP Tunnel
IPSec Tunnel
1620
IPSec
GATEWAY Daemon
Gateway
1610
LAN
1617
Client
1615
Control Path (Thread)
Data Path

FIG. 16B

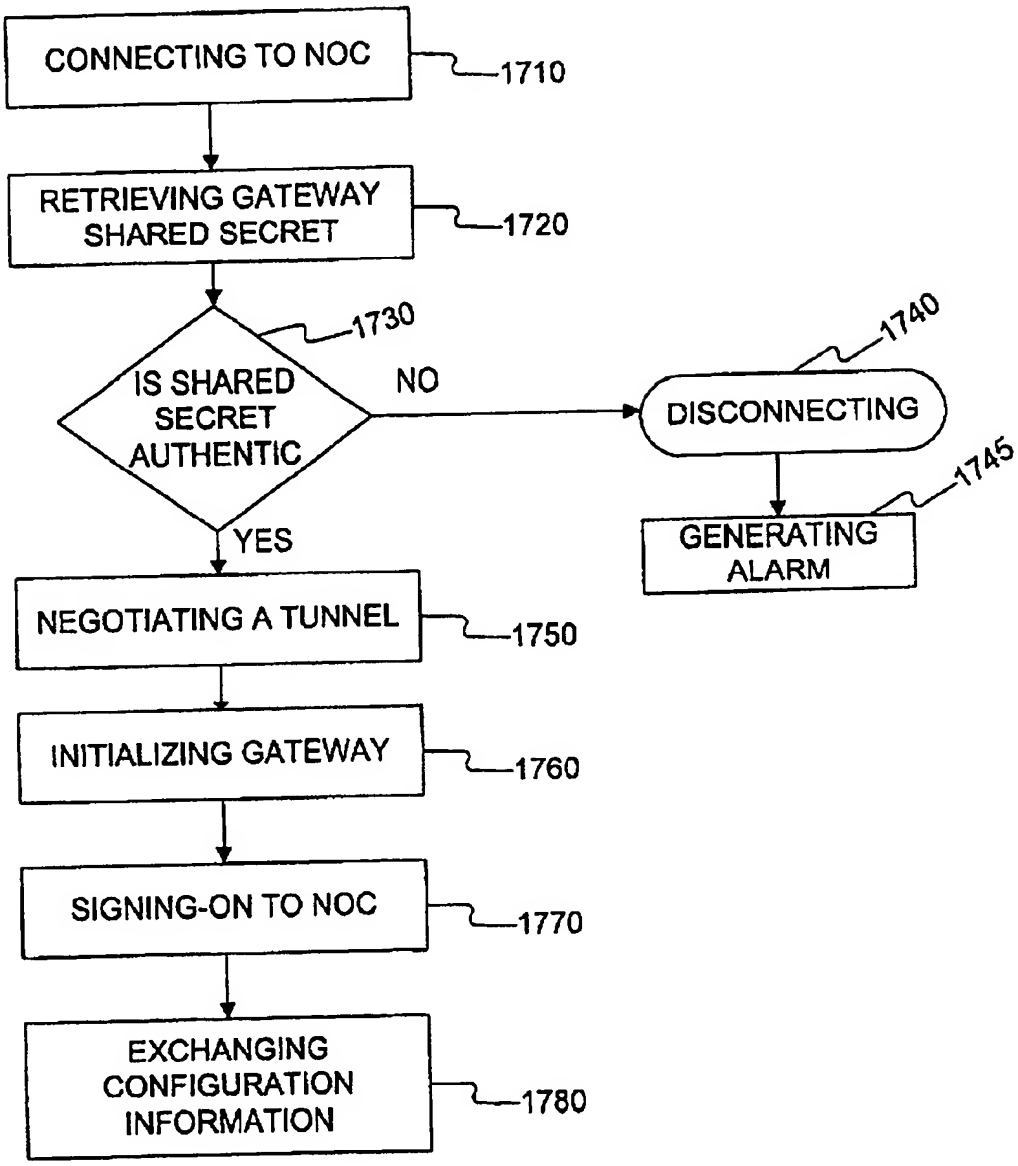

CONNECTING TO NOC
1710
RETRIEVING GATEWAY SHARED SECRET
1720
1730
IS SHARED SECRET AUTHENTIC
NO
1740
DISCONNECTING
1745
GENERATING ALARM
YES
NEGOTIATING A TUNNEL
1750
INITIALIZING GATEWAY
1760
SIGNING-ON TO NOC
1770
EXCHANGING CONFIGURATION INFORMATION
1780

FIG. 17

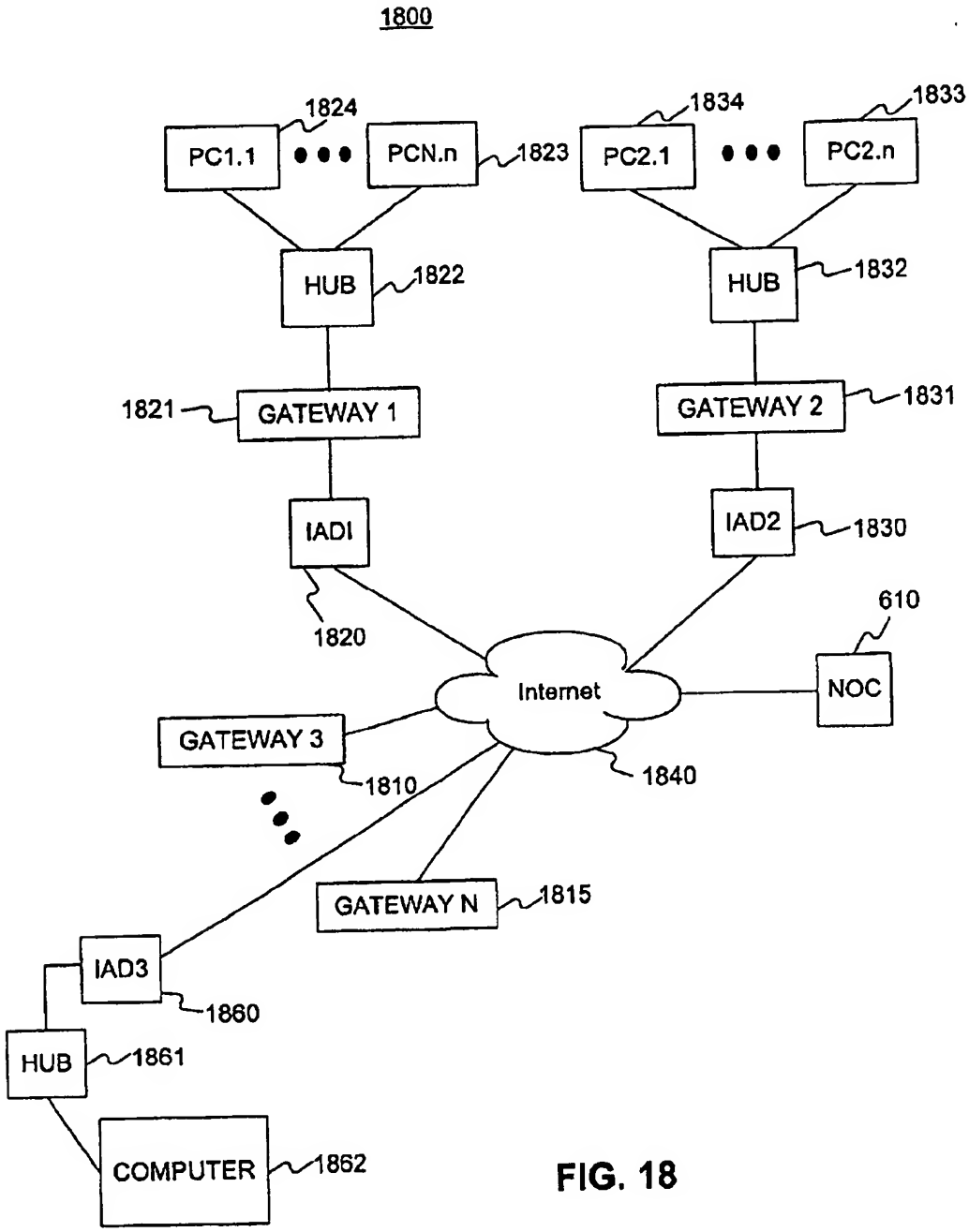

1800
1824
PC1.1
PCN.n
1823
1834
PC2.1
PC2.n
1833
HUB
1822
HUB
1832
1821
GATEWAY 1
GATEWAY 2
1831
IADI
IAD2
1830
1820
610
Internet
NOC
GATEWAY 3
1810
1840
GATEWAY N
1815
IAD3
1860
HUB
1861
COMPUTER
1862

FIG. 18

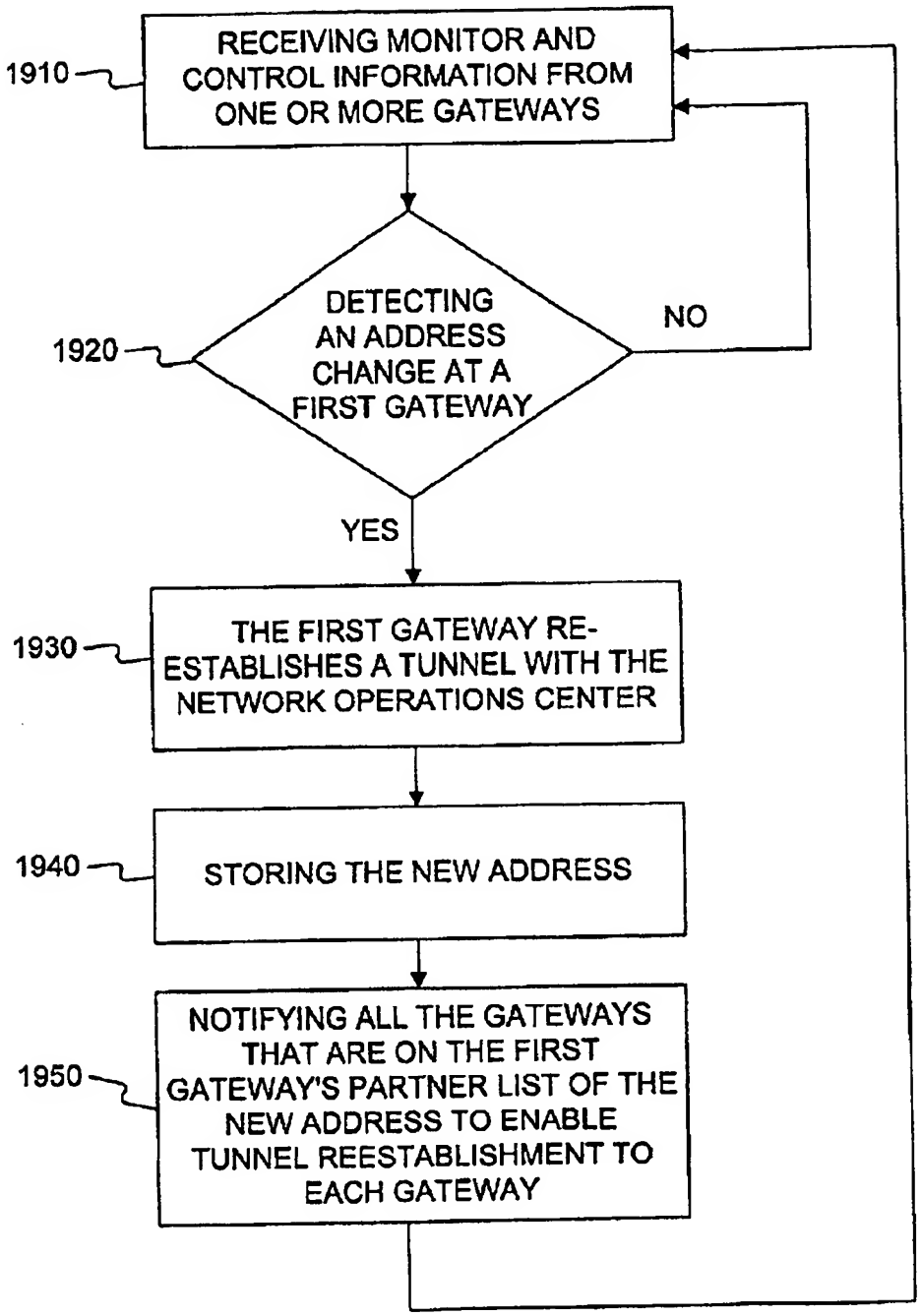

1910
RECEIVING MONITOR AND CONTROL INFORMATION FROM ONE OR MORE GATEWAYS
1920
DETECTING AN ADDRESS CHANGE AT A FIRST GATEWAY
NO
YES
1930
THE FIRST GATEWAY RE-ESTABLISHES A TUNNEL WITH THE NETWORK OPERATIONS CENTER
1940
STORING THE NEW ADDRESS
1950
NOTIFYING ALL THE GATEWAYS THAT ARE ON THE FIRST GATEWAY'S PARTNER LIST OF THE NEW ADDRESS TO ENABLE TUNNEL REESTABLISHMENT TO EACH GATEWAY

FIG. 19

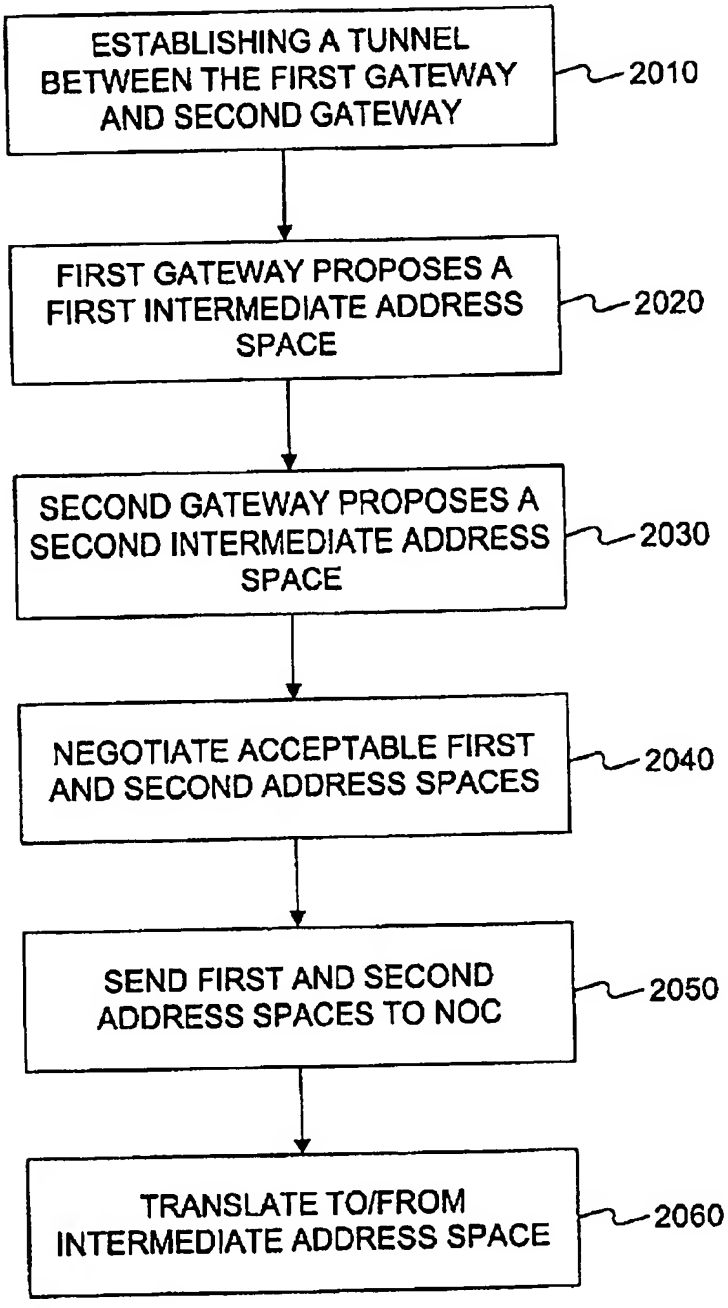

ESTABLISHING A TUNNEL BETWEEN THE FIRST GATEWAY AND SECOND GATEWAY
2010
FIRST GATEWAY PROPOSES A FIRST INTERMEDIATE ADDRESS SPACE
2020
SECOND GATEWAY PROPOSES A SECOND INTERMEDIATE ADDRESS SPACE
2030
NEGOTIATE ACCEPTABLE FIRST AND SECOND ADDRESS SPACES
2040
SEND FIRST AND SECOND ADDRESS SPACES TO NOC
2050
TRANSLATE TO/FROM INTERMEDIATE ADDRESS SPACE
2060

FIG. 20

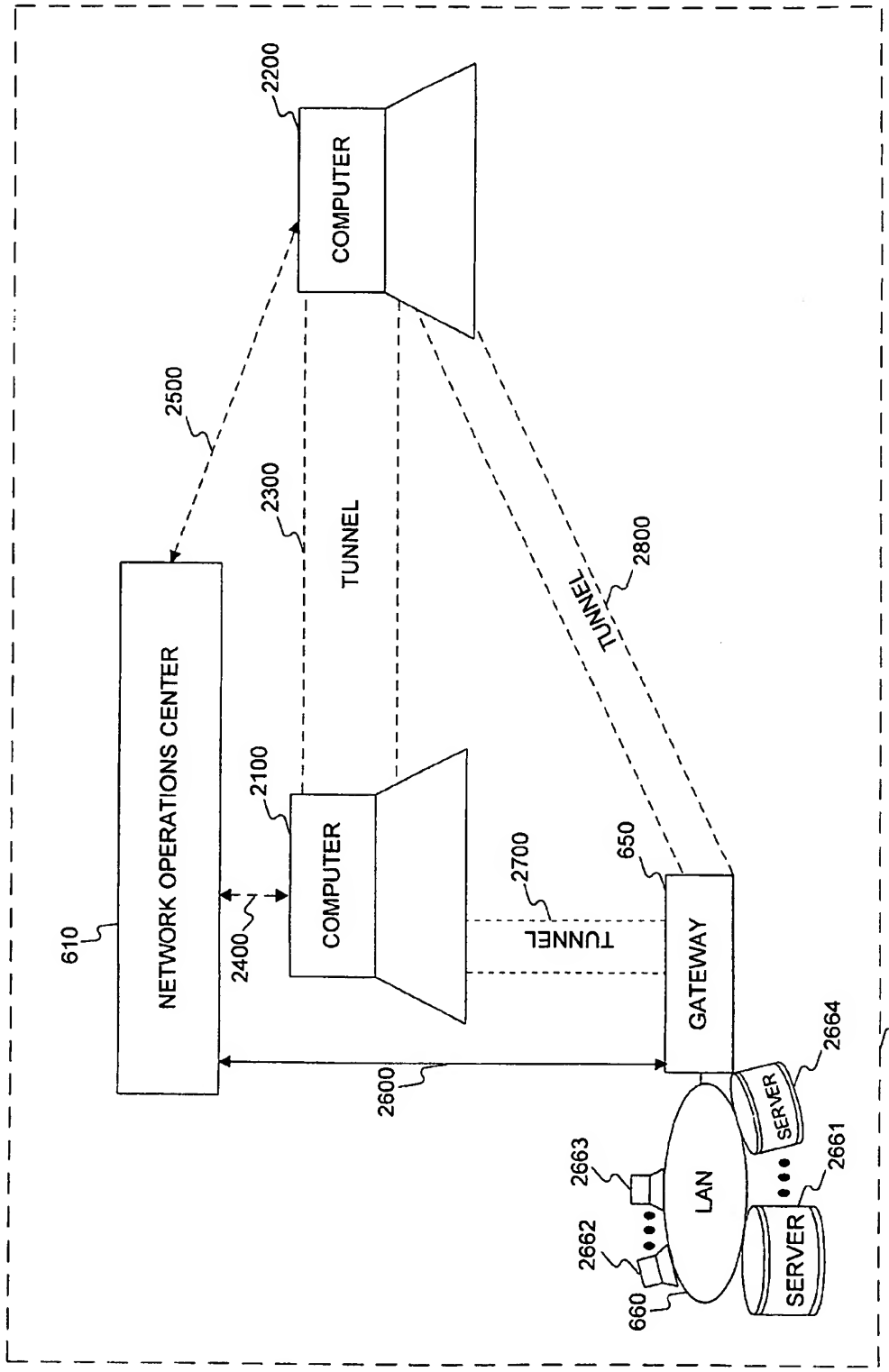

2200
COMPUTER
2500
2300
TUNNEL
2800
TUNNEL
NETWORK OPERATIONS CENTER
610
2100
COMPUTER
2400
2700
TUNNEL
650
GATEWAY
2664
SERVER
2000
2600
2663
LAN
2661
2662
SERVER
660

FIG. 21

METHOD AND SYSTEM FOR MANAGING AND CONFIGURING VIRTUAL PRIVATE NETWORKS

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of U.S. Provisional Patent Application No. 60/196,297, entitled "A METHOD AND SYSTEM FOR MANAGING VIRTUAL PRIVATE NETWORKS," filed on Apr. 12, 2000, the disclosure of which is expressly incorporated herein by reference in its entirety.

DESCRIPTION OF THE INVENTION

[0002] 1. Field of the Invention

[0003] The present invention relates to systems and methods for controlling networks, and in particular, to systems and methods for implementing virtual private networks.

[0004] 2. Background of the Invention

[0005] Wide area networks allow users to access company files and computer programs, regardless of where users are geographically located. Until recently, building wide area networks remained the province of only the largest corporations or companies with enough technical skill and financial resources. Organizations have used a range of approaches to building wide area networks to connect remote offices, partners, or employees. These "traditional" approaches to connectivity include, for example, point-to-point leased lines, packet switched networks, and dedicated virtual private networks (VPNs).

[0006] Point-to-point leased lines are physical networks requiring the engineering of separate links between sites that need to communicate with each other. Point-to-point leased lines can take from 30 to 90 days to install and are costly. A packet switched network using frame relay is a traditional alternative to point-to-point leased lines that offers reduced costs and increased flexibility. Like the point-to-point solutions, the initial installation of a frame relay network takes a long time. For example, additional access circuits may usually take two to three weeks for installation and the service is fairly costly.

[0007] A more-recently introduced service offered by some network service providers is a dedicated virtual private network. This routed service eliminates the complexity and costs associated with the engineering of connections between dedicated locations, but requires the network service provider to manage security as the network is shared with other customers. A virtual private network is "virtual" because it uses a shared or a base network, such as the Internet as its backbone as opposed to a completely private network with dedicated lines. It is also "private" since the information that is exchanged between the users may be encrypted or encoded to provide privacy. Prior to the present invention, virtual private networks, dedicated point-to-point lines, and packet switched networks shared drawbacks of being cumbersome and costly.

[0008] Although traditional virtual private networks offer low access costs, they often entail high set-up, maintenance, and management costs. Based on a number of factors, a shared network such as the Internet has evolved as the preferred backbone for connecting and internetworking multiple locations, partners, and employees. Also, the Internet offers the advantages of being ubiquitous, (available almost everywhere—small towns, large cities, around the world), offering an enormous capacity, and increasing cost-effectiveness, with fast, new access methods, such as DSL and cable modems.

[0009] With the advent and ubiquity of the Internet, virtual private networks have emerged as a way to build a private communication network over a shared public or private infrastructure or a base network. Virtual private networks provide secure private connections over the Internet by enabling authentication of users and locations, delivering secure and private "tunnels" between users or locations, and encrypting user communications.

[0010] Today, most virtual private networks are Internet Protocol (IP) based and are established over the Internet. They fall into two categories, namely hardware-based and software-based virtual private networks. Hardware-based virtual private networks require proprietary hardware platforms and claim to provide high price/performance ratios and potentially increased security through specialized functions. Network manufacturers are building some virtual private network capabilities into routers and other networking equipment.

[0011] Software-based virtual private networks have emerged as another alternative to hardware-based virtual private networks. Vendors are already adding virtual private network functionality, such as tunneling and encryption to their firewall solutions.

[0012] Although use of a base network, such as the Internet as a backbone for wide area networks may be less expensive and more flexible than traditional solutions, the associated costs and complexity of using virtual private networks has been prohibitive. As a result, most companies have been reluctant to link remote locations over the Internet using virtual private networks.

[0013] Building wide area virtual private networks over the Internet has been difficult because most robust solutions have required esoteric networking and security technologies. Merely deciding what type of virtual private network and what levels of security or encryption are required can be confusing to many information technology (IT) personnel and non-IT personnel. Beyond the complex purchase decisions, the installation and ongoing maintenance of such systems can be time-consuming, especially if the number of remote locations changes frequently. In addition, many companies have found that rolling out traditional virtual private network products requires significant logistical planning to make sure that the right hardware and software is available at all the remote locations. Initial configuration of these remote sites is often time consuming enough, without factoring in the effort required to get a remote site back on line if a location fails (especially if no skilled IT resources are available at the remote site).

[0014] Many organizations have been reluctant to establish Internet-based wide area virtual private networks also because of the increasing number of Internet security threats, such as hackers and corporate espionage. Further, virtual private networks and Internet-based connectivity solutions continue to remain prohibitively expensive. Even prepackaged virtual private network solutions require expensive

networking personnel to configure, install, and manage such networks. For example, enterprise level firewall and virtual private network solutions may take up to a week to configure. In addition, the installation often requires support at the remote locations, dictating either extensive travel requirements for home office personnel or the hiring and training of remote IT support staff.

[0015] Many software-based virtual private network solutions also require the purchase of specialized and costly hardware. Moreover, although virtual private networks can save considerable amounts of money over frame relay or leased line networks, associated IT support costs often erase the savings. For example, setting up a virtual private network may necessitate hiring full-time IT professional to set up and administer the network.

[0016] As explained above, the installation and maintenance of a secure virtual private network over the Internet have been too complex, requiring financial investment in hardware, software, personnel, and/or time. To provide encryption and authentication on a virtual private network, each user must perform a variety of tasks including, for example, using an encryption algorithm that is compatible with the virtual private network; using an authentication technique that is compatible with the virtual private network; coordinating various security protocols with other users (e.g., coordinating a public key exchange) of the virtual private network; coordinating the establishment of tunnels with other users of the virtual private network; selecting and manually configuring the encryption path through the communication path; and/or recovering the virtual private network after a failure. Accordingly, the burdens of installing and administering virtual private networks are significant.

SUMMARY OF A FEW ASPECTS THE INVENTION

[0017] To address the above and other limitations of the prior art, methods and systems are provided that easily and effectively leverage the power of a shared or a base network, such as the Internet for private connectivity without the complexity, cost, or time associated with setting up traditional virtual private networks. Rather than requiring specialized hardware, such methods and systems are capable of being self-configured on nonproprietary hardware, such as a standard personal computer (PC), to quickly establish one or more virtual private networks over a local or wide geographical area. Configuration may be achieved by pointing-and-clicking, making it feasible for users to build secure virtual private networks.

[0018] Methods and systems consistent with the present invention enable one or more networks between a first processor and a second processor using at least one additional processor separate from the first and second processors. The additional processor receives information indicating consent on behalf of the first processor to enabling a tunnel between the first processor and the second processor and information indicating consent on behalf of the second processor to enabling a tunnel between the second processor and the first processor. The additional processor determines a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the network. The additional processor provides to each of the first and second processors the first and second virtual addresses to enable one or more tunnels between the first and the second processors, thus enabling one or more networks between the first and second processors.

[0019] Furthermore, methods and systems consistent with the present invention may provide program code that configures a processor, such as the first processor into a gateway capable of being enabled by the additional processor to establish one or more tunnels to another processor, such as the second processor through a communication channel.

[0020] It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the invention, as described. Further features and/or variations may be provided in addition to those set forth herein. For example, the present invention may be directed to various combinations and subcombinations of the disclosed features and/or combinations and subcombinations of several further features disclosed below in the detailed description.

[0021] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate several embodiments of the invention and together with the description, serve to explain the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0022] **FIG. 1** is a general block diagram of a first exemplary network in accordance with methods and systems consistent with the present invention;

[0023] **FIG. 2** is a general block diagram of an exemplary processor in which systems and methods consistent with the present invention may be implemented;

[0024] **FIG. 3** is an exemplary flow chart for initially registering with a control system in accordance with methods and systems consistent with the present invention;

[0025] **FIG. 4** is a general block diagram of a second exemplary network in accordance with methods and systems consistent with the present invention; **FIG. 5** is an exemplary flow chart for establishing a network in accordance with methods and systems consistent with the present invention;

[0026] **FIG. 6A** is a general block diagram of a third exemplary network in accordance with methods and systems consistent with the present invention;

[0027] **FIG. 6B** shows virtual IP addresses for a network in accordance with methods and systems consistent with the present invention;

[0028] **FIG. 7** is an exemplary flow chart for providing information to a Network Operations Center (NOC) in accordance with methods and systems consistent with the present invention;

[0029] **FIG. 8** is an exemplary flow chart for defining a gateway in accordance with methods and systems consistent with the present invention;

[0030] **FIG. 9** is an exemplary flow chart for creating a program code for configuring a processor as a gateway in accordance with methods and systems consistent with the present invention;

[0031] **FIG. 10** is an exemplary flow chart for configuring a processor as a gateway in accordance with methods and systems consistent with the present invention;

[0032] **FIG. 11** illustrates exemplary partner lists in accordance with methods and systems consistent with the present invention;

[0033] **FIG. 12** is an exemplary screen for adding a gateway to the virtual private network in accordance with methods and systems consistent with the present invention;

[0034] **FIG. 13** is an exemplary flow chart for establishing a tunnel in accordance with methods and systems consistent with the present invention;

[0035] **FIG. 14** is a general block diagram of a tunnel between two gateways in accordance with methods and systems consistent with the present invention;

[0036] **FIG. 15** is a general block diagram of two gateways, each not accessible behind a firewall, in accordance with methods and systems consistent with the present invention;

[0037] **FIG. 16A** is a general block diagram of a tunnel between a gateway and a network operations center in accordance with methods and systems consistent with the present invention;

[0038] **FIG. 16B** is a general block diagram of a tunnel between a network operations center and a gateway that includes a client computer in accordance with methods and systems consistent with the present invention;

[0039] **FIG. 17** is an exemplary flow chart for performing the protocol associated with a connection from a gateway to a network operations center in accordance with methods and systems consistent with the present invention;

[0040] **FIG. 18** is a general block diagram of an alternative exemplary network in accordance with methods and systems consistent with the present invention;

[0041] **FIG. 19** is an exemplary flow chart for detecting an address change in a network in accordance with methods and systems consistent with the present invention;

[0042] **FIG. 20** is an exemplary flow chart for resolving address conflicts in a local network in accordance with methods and systems consistent with the present invention; and

[0043] **FIG. 21** is a general block diagram of another exemplary network in accordance with methods and systems consistent with the present invention.

DETAILED DESCRIPTION

[0044] Reference will now be made in detail to the exemplary embodiments of the invention, examples of which are illustrated in the accompanying drawings. Wherever possible, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

[0045] In accordance with an embodiment of the present invention, a prospective user or customer may contact a mediation point or a control system, such as a network operations center via a base network, such as the Internet, and indicate a desire to establish one or more virtual private networks. After answering a series of questions posed by the network operations center, the user receives program code and information for loading onto one or more processors, such as personal computers. The program code and information may be in the form of a disk, such as an optical disk or floppy disk, downloaded over the Internet and onto a disk, or installed directly over the Internet on to a computer. The program code may be distributed to other computers at other desired sites user sites as well. Alternatively, the program code and information may be preinstalled on a computer and delivered to the user.

[0046] The user then runs or boots a computer with the provided code and information. When the computer is booted, it thereafter communicates with the network operations center over the Internet to receive further information such that the computer is configured as a gateway or a computer capable of participating in one or more virtual private networks enabled by the network operations center over a base network, such as the Internet. The provided code and information may also be loaded on other computers such that the computer is configured as a gateway.

[0047] After configuration is completed and based on the user's request, the network operations center may enable over the Internet one or more virtual private networks between the gateway and other gateways configured through the network operations center. At the consent of the user, the virtual private networks may be periodically reconfigured to add additional gateways at, for example, geographically dispersed sites or to provide full or limited access to the networks via other gateways.

[0048] Consequently, the user may configure one or more gateways using a computer, such as a personal computer, without investing in costly proprietary hardware or setting up a typically costly network administration department. Because the gateway as configured is not dependent on a particular piece of hardware, flexible virtual private networks may be inexpensively established between remote locations.

[0049] Accordingly, the user may choose and change its Internet service providers (ISPs), network equipment, and access types (T1, cable modem, DSL, etc.) and then access the network operations center through the Internet to update configuration information that may have resulted from such a change. Furthermore, to participate in a virtual private network, a user need not require other users to use specific network gear or sign-up with specific ISPs. Instead, the user may direct other users to the network operations center to receive program code and information to configure one or more gateways capable of participating in one or more virtual private networks.

[0050] The user may quickly bring up new gateways in minutes rather than weeks or months. As explained above, the user may install the program code, log onto a network operations center with any web browser, and connect to London, New York and Boston in minutes. Unlike traditional virtual private network services requiring 30 to 90 days for installation of a new Internet connection, the gateways may be configured to be compatible with the user's existing Internet connections. The user may even start with a dial-up or ISDN connection and later replace it with a faster DSL, cable, or T1 connection without affecting service. Additionally, unlike traditional network equipment requiring expensive overnight shipping, the gateway pro-

gram code may be downloaded almost anywhere in the world or may be distributed on a storage device, such as an optical disk or a floppy disk.

[0051] In another embodiment, two or more users may register with a controller or network operations center using a web browser. The network operations center may prompt them to provide basic identifying information, such as the Internet Protocol (IP) addresses of their computers. The network operations center may then generate a program code and configuration information and provide them to each user. After the users install the program code and configuration information on their respective computers, the respective computers establish communication with the network operations center to obtain additional configuration information for configuring themselves as gateways. After configuration is completed, one or more of the computers communicates its consent to the network operations center for establishing a tunnel to the other computer. Each computer may communicate its consent mutually and/or independently of the other computer.

[0052] If both gateways consent, the network operations center then proceeds to enable a tunnel between the user computers. The network operations center may enable the tunnel by providing sufficient information to each computer over the Internet such that the computer may establish the tunnel with the provided information. Once the tunnel is enabled, the computers may establish the tunnel and then use the tunnel to exchange information in a secure and trusted manner. At any time, each computer may withdraw its consent and terminate the tunnel. Furthermore, other computers configured through the network operations center may also join the virtual private network.

[0053] Consequently, the tasks of installing a gateway, establishing a virtual private network, and joining a virtual private network are simplified from the perspective of the users, even when establishing a temporary virtual private network for a short term project or a short term financial transaction (e.g., a purchase or sale).

[0054] As such, the described methods and systems may be for various applications, such as, for example, enabling the establishment of virtual private networks without costly hardware and software outlays; providing virtual private networks to businesses that sell products to customers over the Internet; providing virtual private networks to users of a corporate Intranet that seek to share information with outside users in a secure manner; and providing virtual private networks to users of the Internet in general. In such applications, the users may communicate with the virtual private networks by registering over the Internet with a control system, such as a network operations center; installing a program code; and indicating a consent to participate in a virtual private network. As a result, managing virtual private networks is simplified since users are not required to, for example, coordinate selection of encryption algorithms and/or authentication techniques; monitor and/or control tunnels of virtual private networks; and/or recover virtual private networks from failures.

[0055] From a business perspective, the user may be charged a periodic fee based on the number of gateways configured by the user through the network operations center. Alternatively, charges might also be assessed based on one or more of the following: the volume of information transported on the virtual private networks, the number of tunnels, or the usage time.

[0056] Before embarking on an element-by-element description of various preferred embodiments, the following terms are described. A gateway refers to any processor through which access is provided to a network. For example, a gateway may provide hosts or computers in a local area network or in a wide area network access to another network. A processor may include, for example, a personal computer, router, bridge, server, or any other network device. An encrypted information flow includes a flow of information that is encrypted. An example of an encrypted information flow is a tunnel, such as an encrypted tunnel. A tunnel may be established, for example, when two gateways open a channel of communication through a base network, such as the Internet. A tunnel may be enabled, for example, when a gateway is provided with authorization and/or sufficient information that may be used by the gateway to establish a tunnel with another gateway.

[0057] **FIG. 1** shows a general block diagram of a network **100**, in accordance with an embodiment of the present invention. The network **100** may include a control system **175** with one or more network operations centers **170**, a communication channel **120**, one or more gateways **150-153**, one or more local networks **160,161**, one or more hosts **154, 155**, and a computer **101**. The communication channel **120** may include a shared or base network, such as the Internet to facilitate communication and exchanges between the various entities depicted in the network **100** of **FIG. 1**.

[0058] In accordance with an embodiment of the present invention, a first gateway, such as gateway **150** may establish through communication channel **120** a first encrypted information flow to the control system **175**. This first encrypted information flow may permit the control system **175** to exchange control information through the communication channel **120** with the first gateway **150**. Further, a second gateway, such as gateway **151** may establish through communication channel **120** a second encrypted information flow to the control system **175**. This second encrypted information flow may also permit the control system **175** to exchange with the second gateway **151** control information through the communication channel **120**. Since both of these information flows may be encrypted, the encrypted information flow may provide privacy.

[0059] The control system **175** may also enable a third encrypted information flow through the communication channel **120** between the first gateway **150** and the second gateway **151**. The control system **175** may enable the third encrypted information flow after the first gateway **150** and the second gateway **151** consent to enabling the third encrypted information flow.

[0060] The consent communicated to the control system **175** may be mutual in that the first gateway **150** and the second gateway **151** each consents to enabling of the third tunnel. Moreover, the consent may be independent in that the first gateway **150** and the second gateway **151** independently consent to the establishment of the third tunnel without regard to whether the other gateway consents. A gateway may communicate its consent by identifying the names and/or addresses of the other gateways. For example, in an embodiment, a gateway may identify its consent to

enabling a tunnel with another gateway by simply providing the name of the other gateway to the control system **175**. If the control system **175** determines that the consent is mutual (i.e., that the other gateway also consents to enabling the tunnel), the control system **175** places the other gateway on a list (hereinbelow referred to as a partner list) that will be provided to the gateway. Likewise, the control system places the gateway on the partner list for the other gateway. That is, the control system **175** places each gateway on the partner list of the other gateway and provides the respective partner lists to each gateway. Accordingly, the partner list reflects the mutual desire of each gateway to enable a tunnel.

[0061] For example, referring to **FIG. 1**, a user using host computer **155** may use a web browser to access the control system **175** through the tunnel between gateway **150** and the control system **175**. The control system **175** may then provide the user with the names of other gateways that gateway **150** may establish a tunnel with (e.g., the names for gateways **151-153**). The user then may select one or more names corresponding to the other gateways that gateway **150** consents to enabling a tunnel with. The user may then submit the names of the selected gateways to the control system **175**, which determines if there is mutual consent for each of the selected gateways. That is, the control system **175** determines for each of the selected gateways whether or not the selected gateway also consents to enabling a tunnel with gateway **150**. If there is mutual consent, each of the selected gateways that also consents is added to the partner list for gateway **150**, and gateway **150** is also added to the partner list for each of the selected gateways. These partner lists may then be forwarded by the control system **175** to gateway **150** and each of the selected gateways.

[0062] Accordingly, when the control system **175** determines that the first gateway **150** and the second gateway mutually consent to the third tunnel, the control system may then provide to the first and second gateways through the first and second tunnels, respectively, sufficient information to enable the third tunnel. The third tunnel may be enabled, for example, when the first and second gateways are provided sufficient information allowing them to establish this third tunnel through the communication channel **120**. In one embodiment, the sufficient information includes the partner list for the first gateway and the partner list for the second gateway. Moreover, for each gateway listed on the partner list, the partner list may include, for example, a virtual IP address, a real IP address, and/or other information describing each gateway. After the third tunnel is enabled, the first and second gateways **150, 151** may establish the third tunnel through the communication channel **120**. This third tunnel may provide privacy as to the exchanged information and may also be authenticated using an Internet Protocol Security (IPSec) compliant authentication technique, such as MD-5 hashing. Also, the encryption used for the encrypted information flow may be a weak encryption or encoding algorithm that provides minimal privacy or may be a strong encryption scheme that essentially guarantees privacy.

[0063] An encrypted information flow, such as a tunnel may be established through communication channel **120** by, for example, encapsulating a protocol within another protocol. For example, a tunnel may be encrypted when an Internet Protocol packet encapsulates an encryption protocol. Examples of encryption protocols may include RSA, Digital Encryption Standard (DES), and Triple DES (3DES). For example, an encrypted tunnel may be established using Internet Protocol (IP) packets such that the payload of each packet is encrypted but the address of each packet is unencrypted (i.e., clear-text). As a result, the encrypted payload may be encapsulated by a clear text IP address, forming a virtual tunnel through a base network, such as the communication channel **120**. Other encrypted tunnels may be established through the communication channel **120** with other gateways, such as gateways **152** and **153**. These virtual tunnels established through the base network and enabled by the control system **175** may also form a virtual network. If a virtual network enabled by the control system **175** uses some type of encoding or encryption for privacy, the virtual network may also be referred to as a virtual private network.

[0064] In the embodiment of **FIG. 1**, the computer **101** may include, for example, a personal computer and/or a workstation that include a web browser, such as the Netscape Navigator developed by Netscape or the Internet Explorer developed by Microsoft. The computer **101** may connect to the control system **175** through the communication channel **120** using the web browser. Once the computer **101** connects to the control system **175**, a user may register one or more gateways with the control system **175** and define an initial configuration for one or more of the gateways **150-153** desiring to participate in one or more virtual private networks.

[0065] After the initial configuration of the gateways **150-153** is defined, the control system **175** may create a disk image that includes program code and information for configuring the gateways **151-153**. The disk image may include, for example, a copy of the program code required to configure a personal computer as a gateway. Alternatively, the control system **175** may install through the communication channel **120** a bootable program on the gateways **151-153**. After executing the bootable program on a computer, the bootable program may retrieve additional program code and configuration information from the control system **175** or other secured site to configure the computer as a gateway. Moreover, the program code may be loaded onto the gateways **150-153** using a single disk (not shown) and/or downloaded through the communication channel **120**. Once the program code is installed, the gateways **150-153** may be capable of being enabled by the control system **175** and participating in one or more virtual networks or virtual private networks through the communication channel **120**.

[0066] The disk image may include program code for one or more of the following: program code for IPSec; program code for communications between network operations center **170** and gateways **151-153**; the Linux Operating System (OS) including kernel and device drivers; the configuration of the IP stack such as a Dynamic Host Configuration Protocol (DHCP) client and a DHCP Server; program code for routing packets through one or more tunnels established between gateways **151-153**; access control information for limiting the functions performed through one or more tunnels established between gateways **151-153**; program code for the SOCKS Proxy code; program code for a web browser; and any other software that may be installed based on the user's configuration. In addition, the LINUX operating system may be a "hardened" version of Linux to improve the security of the operating system. When each of the gateways **150-153** loads the disk image, each gateway

may execute the program code contained in the disk image. As each of the gateways **151-153** performs the steps contained in the program code, each may connect to the control system **175** and establish an encrypted information flow to the control system **175**.

[0067] The control system **175** may also enable an encrypted information flow between at least two gateways, permitting them to exchange information or traffic in a private manner. Further, the control system **175** may control and/or monitor the encrypted information flows in the network **100** by exchanging control and/or monitoring information with the gateways over the encrypted information flow.

[0068] Referring to **FIG. 1**, the control system **175** may include one or more network operation centers **170**. Each of the network operation centers **170** may be located at the same location or may be distributed along the communication channel **120** connecting the distributed network operation centers **170**. If the network operations centers **170** are distributed, they may also use one or more gateways configured as described above to provide privacy and/or authentication. The control system **175** and the network operation centers **170** may be implemented with at least one processor including, for example, one or more of the following components: a central processing unit, a co-processor, a memory, a storage device, an input device, an output device, a network interface, a display, and/or other processing devices and systems.

[0069] The gateways **150-153** may each include, for example, one or more of the following processors: a computer, a server, a router, a switch, a portable device such as a cell phone or a personal digital assistant, or any other communication device capable of performing the functions of the gateway in accordance with the present invention. A gateway may participate as a stand-alone node or computer interfacing the communication channel **120** (see, e.g., the gateways **152** and **153**) and/or as a gateway interfacing a local network (see, e.g., the gateways **150** and **151**). In a stand-alone configuration, for example, the gateway **153** may permit a user to participate in one or more virtual private networks established over communication channel **120**. In a local network configuration, for example, the gateway **150** may interface the local network **100** to permit one or more users, such as hosts **154** and **155** to participate in one or more virtual private networks established over communication channel **120**. Furthermore, in the local network configuration, the gateway may resolve address conflicts that may exist with the local area network **160** and other networks such as local area network **161**.

[0070] The host computers **154** and **155** may each include a processor, such as a computer **200** shown in **FIG. 2**. The computer **200** may include an input module **205**, a central processing unit (CPU) **220**, a storage module **250**, and an output module **230**. The output module **230** may include a display **235**, a printer **236**, and a network interface **238**. One of ordinary skill in the art will recognize that each host computer **154** and **155** may also function as a gateway in accordance with the present invention. Although **FIG. 2** shows a computer **200**, other devices, such as printers, personal digital assistants, wireless devices, and mobile phones, may function as a host computer and participate in one or more virtual private networks established over communication channel **120**.

[0071] The input module **205** of **FIG. 2** may be implemented with a variety of devices to receive a user's input and/or provide the input to the CPU **220**. Some of these devices (not shown) may include, for example, a network interface module, a modem, a keyboard, a mouse, and an input storage device.

[0072] Although **FIG. 2** illustrates only a single CPU **220**, computer **200** may alternatively include a set of CPU. The CPU **220** may also include, for example, one or more of the following: a co-processor, memory, registers, and other processing devices and systems as appropriate.

[0073] The storage module **250** may be embodied with a variety of components or subsystems including, for example, a hard drive, an optical drive, a general-purpose storage device, a removable storage device, and/or other devices capable of storing. Further, although storage module **250** is illustrated in **FIG. 2** as being separate or independent from CPU **220**, the storage module and CPU **220** may be implemented as part of a single platform or system.

[0074] Referring again to **FIG. 1**, the communication channel **120** may facilitate communication between the various entities depicted in the network **100**. The communication channel may include, for example, a telephony-based network, a local area network (LAN), a wide area network (WAN), a dedicated Intranet, the Internet, and/or a wireless network. Further, any suitable combination of wired and/or wireless components and systems may be incorporated into the communication channel **120**. Any suitable combination of point-to-point communications or network communications may also be incorporated into communication channel **120** to facilitate communication between the entities illustrated in **FIG. 1**. Moreover, although local networks **160,161** are shown as being separate from the communication channel **120**, the local network **160,161** may be implemented in the same manner as the communication channel **120** or include one or more of the features of the communication channel **120**.

[0075] In one embodiment, a user may serve as an administrator and may register at least one of the gateways **150-153** through control system **175** and/or establish one or more virtual private networks over communication channel **120**. The user may use an Internet browser on computer **101** to contact the control system **175**, to register at least one of the gateways **150-153**, and/or establish one or more virtual private networks over communication channel **120**. Moreover, although the computer **101** is shown as a stand-alone entity in the embodiment of **FIG. 1**, the computer **101** may alternatively be co-located with one or more of the gateways **150-153**, the control system **170**, and/or the communication channel **120**.

[0076] Furthermore, the user may register with the control system **175** and provide basic information, such as the number of gateways participating in the virtual private network and billing information. Once registered, the user may receive code generated by the control system **175**. The user may then reboot a computer with the received code to configure the computer as a gateway. That is, the administrator may install the code on any computer that the administrator desires to configure as a gateway including the computer serving as the computer **101**. The configured gateway may then establish a tunnel to another gateway (i.e., similarly configured by the control system **175**) after the

control system **175** determines that each gateway mutually consents to enabling the tunnel and provides each gateway with sufficient information to enable the tunnel.

[0077] **FIG. 3** shows an exemplary flowchart for initially registering one or more gateways with the control system **175**. Referring to **FIGS. 1 and 3**, the user may register at least one of the gateways **150-153** with the control system **175** (step **310**) and define a configuration for the registered gateways **150-153** (step **320**). In one embodiment, the user may contact the control system **175** through the Internet using a web browser to specify a particular configuration for a gateway. This specified configuration information may include a name for the gateway and a name for the virtual private network. This name for the virtual private network will hereinafter be referred to as the virtual private network's domain name.

[0078] The control system **175** may use the specified configuration to assemble code and information, such as program code and textual information (e.g., Extensible Markup Language also referred to as "XML"), in the form of a disk image (step **330**). This disk image may include all the program code and information needed to configure gateways **150-153** for establishing one or more virtual private networks established over communication channel **120**. The disk image may then be provided to the user and installed on a processor, such as a personal computer or a general-purpose computer (step **340**). When the processor reboots, it uses the information provided in the disk image to configure itself as a gateway capable of establishing secure tunnels to the control system **175**. The disk image may be sized to fit on a single storage medium, such as a floppy disk or optical disk. Moreover, the disk may be distributed through alternative channels of distribution, such as direct mail, unsolicited mail, over-the-counter retail, or may be distributed with other hardware and software provided by a vendor. Alternatively, the disk image may be downloaded from the control system onto a storage medium or may be stored at the control system **175** for later transfer to the gateways **150-153**. Accordingly, a commercial-off-the-shelf computer may be configured as a gateway capable of participating in one or more virtual private networks established over communication channel **120**.

[0079] The control system **175** may perform various functions including, for example, enabling tunnels between two or more gateways in network **100**; assembling and/or configuring a user's computer as a gateway; negotiating an authentication technique; determining one or more partner lists for the gateways **150-153**; administering the configuration of virtual private networks established over communication channel **120**; providing virtual Internet Protocol (IP) addresses to each gateway; monitoring and/or controlling the established virtual private networks; enabling the establishment of tunnels between two or more gateways in the network **100**; enabling the establishment of tunnels with gateways not accessible behind firewalls; and/or recovering the established virtual private networks after a failure. The control system **175** may exchange control information with each of the gateways **150-153** through a tunnel established through the communication channel **120**. Moreover, each pair of the gateways **150-153** may exchange information through one or more tunnels established between the gateways.

[0080] **FIG. 4** shows an exemplary virtual private network **400** established over the communication channel **120**. This exemplary network **400** will be used to illustrate how such a network is enabled. The network **400** includes a first gateway **450**, a second gateway **451**, a computer **401**, a first tunnel **425**, a second tunnel **426**, a third tunnel **423**, and the control system **175**. The first tunnel **425**, the second tunnel **426**, and the third tunnel **423** may be established through the communication channel **120**. Moreover, gateway **450** and gateway **451** may each participate as a stand-alone node in the virtual private network **400** or as a node interfacing a local network, such as local network **160** shown in **FIG. 1**.

[0081] The virtual private network **400** may be established after each of the gateways **450, 451** establishes a tunnel (e.g., the first tunnel **425** and the second tunnel **426**) to the control system **175**; after the first gateway **450** and the second gateway **451** each communicate to the control system **175** a consent to enable the third tunnel **423** between the first gateway **450** and the second gateway **451**; after the control system **175** provides to the first gateway and the second gateway sufficient information to enable the third tunnel **423**; and after the first gateway **450** and the second gateway **451** establish the third tunnel **423**. With the third tunnel established, the first gateway **450** and the second gateway **451** may communicate in a private and/or trusted manner. Although **FIG. 4** only shows two gateways, additional gateways (not shown) may also join the virtual private network **400**. Accordingly, the task of configuring gateways that are capable of participating in a virtual private network is significantly simplified.

[0082] A user desiring to configure the virtual private network **400** may simply register one or more gateways and administer the network through the control system **175**. The tasks performed by the user may thus be simplified to, for example, initially registering with the control system, rebooting one or more computers with software provided by the control system to configure the computers as gateways, and selecting one or more gateways from a list of desired partners. When two gateways consent to enabling a tunnel between the two gateways, the control system **175** may place each gateway on the partner list of the other gateway and provide the partner list to each gateway. Accordingly, the partner list may reflect the mutual desire of each gateway to enable a tunnel.

[0083] Moreover, the control system **175** may perform at least one or more of the following tasks, which are otherwise typically administered by the users enabling tunnels between gateways; coordinating one or more partner lists; administering the configuration of one or more virtual private networks established based on the enabled tunnels; monitoring the virtual private networks; controlling the virtual private networks; distributing to gateways information about changes in the configuration of the virtual private networks and/or other gateways; disseminating software for configuring gateways; providing an indication of a compromised private key; negotiating an encryption algorithm with gateways; negotiating an authentication technique with gateways; and recovering from a failure in the virtual private networks.

[0084] As previously discussed with reference to **FIG. 3**, after a user desiring virtual private network services registers for secure services, the control system may assemble a

disk image and provide the disk image to the user for loading onto a computer and configuring the computer as a gateway. The gateway may then participate in a virtual private network established over a base network, such as the Internet.

[0085] **FIG. 5** illustrates an exemplary flow chart of the steps for establishing a virtual private network between the gateways identified by the user. Each of these steps will be discussed in further detail following the broad description of **FIG. 5**.

[0086] Referring to **FIGS. 4 and 5**, the first gateway **450** may start with the disk image installed (step **510**). The first gateway **450** may establish a connection to the control system **175** (step **520**) and proceed to establish a first tunnel **425** to the control system **175** (step **530**) through a communication channel, such as the communication channel **120** of **FIG. 1**. The second gateway **451** may also perform the steps **510-530** to establish a second tunnel **426** to the control system **175**. Once the first and second tunnels are established, the control system **175** may exchange information with each gateway to further configure the gateways.

[0087] To enable a third tunnel **423** between the first gateway **450** and the second gateway **451** (step **540**), the control system **175** may determine whether the first gateway **450** and the second gateway **451** have consented to enabling the third tunnel **423**. This consent may be mutual and independent of the decision of the other gateways (not shown). For example, the control system **175** may determine the consent based on a list that includes desired partners for each of the gateways **450, 451**. If the first gateway **450** and the second gateway **451** each consent to enabling of the third tunnel **423**, the control system **175** may then enable the third tunnel **423** (step **540**).

[0088] For example, to enable the third tunnel (step **540**), the control system **175** may perform one or more of the following: update the partner lists of the first gateway **450** and the second gateway **451** to reflect mutual consent; provide an indication that a tunnel between the first and second gateways **450, 451** is authorized; provide real IP addresses for each of the gateways to permit a connection through a base network, such as the Internet; provide the virtual IP address of each gateway to the other gateway to enable a tunnel between the gateways; facilitate the establishment of one or more tunnels by providing out-of-band signaling to the first gateway **450** and the second gateway **451** through the first tunnel **425** and the second tunnel **426**, respectively; determine one or more partner lists for one or more gateways **450, 451**; provide configuration information for the network and/or for each gateway; exchange control information with the first gateway **450** and the second gateway **451** on the first tunnel **425** and the second tunnel **426**, respectively; negotiate an encryption algorithm with each gateway; and negotiate an authentication technique. Moreover, the control system **175** may also monitor the status and performance of the tunnels established through the communication channel **120** (step **550**).

[0089] **FIG. 6A** shows a third exemplary network **600** in accordance with an embodiment of the present invention. The network **600** may include one or more local area networks (LANs) **660, 661**, a first, second, and third gateways **650-652**, the Internet **620** and/or Intranet access (not shown), and a network operations center **610**.

[0090] The LANs **660, 661** may be similar to the LANs **160,161** of **FIG. 1**. The Internet **620** and/or Intranet access may include features similar to the communication channel **120** of **FIG. 1**. Moreover, the gateways **650-652** may each include information and program code for implementing one or more virtual private networks over the Internet **620**. Furthermore, the first and second gateways **650, 651** may interface the LAN **660, 661** and the network **600** whereas the third gateway **652** may be configured as a stand-alone node interfacing only the network **600**.

[0091] In the embodiment of **FIG. 6A**, the network operations center **610** may determine a virtual address for each gateway desiring to participate in one or more virtual private networks established through a base network, such as the Internet **620**. Consequently, each gateway may be provided two addresses—a real or public address and a virtual address. The virtual address, which may be in an IP format, may be used by the gateways to establish one or more tunnels with each other through a base network, such as the Internet **620** and may be routable only through the established tunnels. This virtualized addressing may provide virtual connectivity through the Internet **620** and may allow routing of virtual addresses from one address to another. Moreover, this virtualized addressing may facilitate network address translation, port address translation, IP masquerade, and/or IP connection sharing during the process of routing as well as during the dynamic assignment of addresses. Although a virtual address may be used by a gateway to establish one or more tunnels to form a virtual network and/or virtual private network, the network operations center **610** may alternatively provide to each gateway any other address that is capable of enabling any other networks established through or over a base network, such as the Internet **620**.

[0092] Based on the virtual addresses determined by the network operations center **610** and provided to the gateways **650, 651, 652**, one or more virtual private networks may be established over the Internet **620**. For example, each gateway **650, 651, 652** may include a virtual device adapter (not shown), which may be capable of emulating the functions of a network interface card (NIC). Using the virtual device adapter, each gateway may route or forward information, such as packets through tunnels established with other gateways.

[0093] **FIG. 6B** shows the network **600** of **FIG. 6A** from the perspective of virtual addresses and real or public addresses that are used by gateways **650-652** to route information, such as packets through tunnels established through the Internet **620**, in accordance with an embodiment of the present invention. The gateways **650-652** may be assigned real IP addresses **601, 602, 603** and virtual IP addresses **604, 605, 606**, respectively. Each real IP address, which may be assigned by, for example, an Internet Service Provider (ISP), may be routable through a base network, such as the Internet **620**. On other hand, each virtual address, which may be assigned and provided by the network operations center **610**, may be only routable through the tunnels enabled by the network operations center **610** and established through the Internet **620**.

[0094] The solid lines connecting the gateways **650-652** represent the real IP connectivity between the machines. The real IP addresses **601-603** used by gateways **650-652**,

respectively, may interface the Internet **620** or a local area network, such as LAN's **660** and **661**. The dashed lines represent virtual connectivity provided by the virtual IP addresses **604-606**. Each gateway may include at least one virtual device adapter with a corresponding virtual IP address. For example, a virtual device adapter (not shown) may be included at each end of a tunnel **699** established between the first gateway **650** and the second gateway **651**. Each virtual device adapter may have the corresponding virtual IP address for its gateway. For example, the virtual device adapter for the first gateway **650** may have a virtual IP address of 10.0.1.1 (shown as **604**), and the virtual device adapter for the second gateway **651** may have a virtual IP address of 10.0.1.2 (shown as **605**).

[0095] In one embodiment, the network operations center **610** may provide to each gateway a virtual IP address during the initial configuration of the gateway. The network operations center **610** may then store the virtual IP address of the gateway with the gateway's name and the authentication information, such as a shared secret for that gateway. To enable a tunnel between two gateways that mutually consent to the tunnel, the network operations center **610** may provide each gateway the virtual IP address of the other gateway.

[0096] Packets addressed with a virtual IP address may be transported between the gateways through tunnels established through a base network, such as the Internet **620**. For example, when a pair of gateways (e.g., **650** and **651**) consents to enabling a tunnel (e.g. tunnel **699**) between the gateways, the network operations center **610** may provide the virtual addresses for each gateway to the other gateway to enable the tunnel between the gateways.

[0097] Before the first gateway **650** sends a packet with an encrypted payload through a tunnel to the second gateway **651**, the virtual device adapter may add the virtual addresses of the second gateway **651** and the first gateway **650** to the packet. For example, the virtual device adapter may add a source virtual address of 10.0.1.1 (shown as **604**) and a destination virtual address of 10.0.1.2 (shown as **605**) to a packet from the first gateway **650** to the second gateway **651**. The first gateway **650** may then take the virtualized packet and encapsulate the virtualized packet within another TCP/IP packet with real source and destination addresses, such as a source address of 193.168.100.5 (shown as **601**) for first gateway **650** and a destination address of 193.11.10.3 (shown as **602**) for second gateway **651**. The encapsulated packet may then be routed based on the real destination address of 193.11.10.3 through the Internet **620** until the packet reaches the real destination address.

[0098] When the encapsulated packet arrives at the destination address, the second gateway **651** may remove the real TCP/IP addresses, leaving a payload that includes an IP packet with the virtual source and destination addresses. The virtual device adapter within the second gateway **651** may recognize the virtual IP addresses, receive the packet with the virtual IP addresses (i.e., source and destination virtual addresses), and forward the packet to the second gateway **651** for additional processing, such as authenticating and/or decoding the encrypted payload of the packet.

[0099] In one embodiment, network operations center **610** may enable and administer one or more virtual private networks, such as tunnels established through the Internet **620**. The network operations center **610** may include one or more processors that are distributed or co-located within substantially the same geographic area. For example, the network operations center **610** may be distributed along a communication channel (see, e.g., the communication channel **120** at **FIG. 1**), the Internet, and/or an Intranet.

[0100] The network operations center **610** may perform at least one or more of the following features: providing information and code for configuring processors, such as computers as gateways capable of participating in one or more virtual private networks established through the Internet **620**; enabling the establishment of tunnels by providing an indication that a tunnel between two gateways is authorized; determining one or more partner lists for gateways; administering the configuration of the virtual private networks; detecting and resolving virtual and real IP address conflicts; monitoring the virtual private networks; controlling the virtual private networks; negotiating an encryption algorithm with each of the gateways; providing a virtual IP address to each gateway; negotiating an authentication technique with each of the gateways; distributing changes to the configuration of the virtual private network; disseminating software updates to the gateways; providing an indication of a security problem (e.g., a compromised private key); and recovering the virtual private networks from failures.

[0101] Accordingly, a user's role is simplified to registering with the network operations center **610**, providing configuration information about one or more of the desired gateways, loading program code onto one or more computers to configure them as gateways, and selecting one or more desired partners for establishing one or more virtual private networks over a base network, such as the Internet **620**.

[0102] Referring back to **FIG. 6A**, the network operations center **610** may include a public web server **611**, a tunnel interface module **612**, a proxy module **613**, a controller module **614**, an administrative server **615**, a database server **616**, one or more firewalls **617**, one or more switches **680**, and a communication channel **681**.

[0103] The public web server **611** may not authenticate the identity of those connected to the public web server **611**, and thus, may not provide any measure of trust. Moreover, the public web server **611** may not provide encryption or privacy. But the public web server **611** may provide a user with a means of accessing the network operations center **610** to perform limited functions, including registering to enable and establish a virtual private network through the Internet **620**.

[0104] For example, a user may register through the public web server **611** in a nonsecure manner. During initial registration, the network operations center **610** and/or the public web server **611** may present to the user a series of questions and receive responses to the question based on which the network operations center **610** may generate program code and information for configuring a computer as a gateway capable of participating in one or more virtual private networks established over the Internet **620**. For example, this program code and information may be provided in the form of a disk image, which may be downloaded and installed in one or more computers to configure them as gateways **650-652**. Moreover, the public web server **611** may also include one or more of the following: marketing information, trouble ticket information, and other user information that may not require privacy and/or authentication.

The public web server **611** may include a firewall **617** and other security devices to limit access to the switch **680** and the communication channel **681** in network operation center **610**. In one embodiment, the Linux Ipchains utility may be used to manage the firewall **617**.

[0105] The tunnel interface module **612** may include program code for establishing tunnels between the network operations center **610** and one or more of the gateways **650-652**. The tunnel interface module **612** may also include a public addressable or routable IP address that permits establishing tunnels between the network operations center **610** and the gateways **650-652** through the Internet **620**. Moreover, the tunnel interface module **612** may include a transmission control protocol (TCP) tunnel driver used to establish a TCP tunnel between the network operations center **610** and the gateways **650-652**. For example, the tunnel interface module **612** may use the TCP tunnel driver to encapsulate packets for an IPSec tunnel within TCP packets. Although the TCP tunnel driver may encapsulate the IPSec tunnel, other encryption and/or tunnel software (e.g., a User Datagram Protocol (UDP) tunnel driver) may be used instead.

[0106] In one embodiment, the only processes that may be executed from the nonsecure side of the tunnel interface module **612** (i.e., the Internet side **620**) may be those processes related to the TCP tunnel driver.

[0107] To enhance security, the tunnel interface module **612** may communicate with the other subsystems of the network operations center **610** in a limited manner. For example, the tunnel interface module **612** may provide a single control and monitoring port for exchanging messages with the controller module **614** and for exchanging secured sockets layer (SSL) messages with the administrative server **615**. Further, the tunnel interface module **612** may use a firewall **617** and/or other security devices to limit access to the switch **680** and communication channel **681**. The two-tier structure with the tunnel interface module **612** connected through security devices, such as firewalls to the controller module **614** may provide enhanced security at the network operations center **610**.

[0108] The proxy module **613** may include one or more processors, which may serve as a proxy for enabling one or more tunnels between at least two of the gateways **650-652**, when the gateways are each not accessible behind a firewall, hiding their respective real IP addresses. Alternatively, the proxy module **620** may be located within one of the gateways **650-652** or at a third party website hosting the proxy module **613**.

[0109] The controller module **614** may include one or more processors, which may receive the control information provided by each of the gateways **650-652**. The control information provided by each of the gateways **650-652** may also include monitoring information. The controller module **614** may also authenticate the identity of a gateway, determine that tunnels are authorized according to each gateway's list of desired partners, and add partners to each gateway's partner list.

[0110] The administrative server **615** gathers information and then may store gathered information in the database server **616** including, for example, a tunnel database that includes a list of tunnels that are active on the network **600**; a predefined rule or trigger that indicates when a new tunnel request is made for a tunnel that already exists and is active in the tunnel database; a database with authentication information capable of authenticating the identity of each of the gateways **650-652** participating in the network **600**. For example, the database server **616** may store for each gateway the authentication information in the form of a shared secret (e.g., a bit string and/or a public key) that authenticates the identity of a gateway seeking to establish a tunnel to the network operations center or another gateway. When the shared secret stored in the database server **616** matches the shared secret presented by the gateway to the network operations center **610**, the gateway may be authenticated.

[0111] While encryption techniques may make communications private, authentication techniques may allow communicating parties to verify each other's identity and the authenticity of the exchanged information. Authentication serves to provide a level of trust so that users in a virtual private network may be confident about the authenticity of the exchanged information. Authentication may be established using a variety of security techniques including, for example, a signature, a digital signature, a digital certificate, a hash code, a password, and/or any other approach that may be used to establish identity of a user or computer.

[0112] The database server **616** may perform one or more of the following: storing customer information; storing the disk image described above; generating reports, such as alarm reports, activity reports, and/or other reports for administering virtual private networks established through the Internet **620**; and storing monitoring information associated with the virtual private networks.

[0113] The firewalls **617** may include one or more processors which may selectively limit the type of information reaching communication channel **681** and switch **680**. For example, the firewalls **617** may only permit entry of TCP commands to a specific port number. Moreover, the firewalls **617** may be implemented as a stand-alone device, software, firmware, and/or implemented as part of another processor, router, gateway, and/or any other device capable of performing the functions of a firewall.

[0114] The switches **680** switch information or traffic (e.g., datagrams, packets, or cells) between one or more of the subsystems **611-616** of the network operations center **610**. The switches **680** may be implemented with one or more processors, a router, a switch, and/or any other communication device capable of switching and/or routing information to the appropriate subsystem within the network operations center **610**.

[0115] The subsystems **611-616** of the network operations center **610** may be distributed along the communication channel **681** that connects the subsystems. The communication channel **681** may include one or more of the features and functions described above with respect to the communication channel **120** of **FIG. 1**.

[0116] **FIG. 7** shows a flowchart of the steps performed for registering a gateway. A user, such as an administrator may register a gateway with the network operations center **610**. A computer may connect through a gateway **650** to the Internet **620** and the public web server **611** of the network operations center **610** (step **710**). Alternatively, a computer may connect directly to the Internet **620** and the public web

server **611**. The user of the computer, who may function as an administrator of the gateway **650**, may provide registration information (step **720**) to the public web server **611**. The public web server **611** may then store the registration information (step **730**) in, for example, the database server **616**. The initial registration information may include preliminary configuration information, such as the number of gateways, billing information, and the administrator's name and email address.

[0117] Since the initial connection between the user's computer and the network operations center **610** may be a nonsecure connection, it may be desirable to limit the initial registration information to a minimum (e.g., the registration information provided above in step **720**) to enhance security. This initial registration information may include the minimum amount necessary to create program code and information needed to configure a processor such that the configured processor is capable of contacting the network operations center **610** over a secure connection (e.g., a tunnel) established over the Internet **620** to obtain additional configuration information. Accordingly, once the user is able to communicate with the network operations center **610** through the secure connection, the user may then provide additional registration information. This additional information may be needed to complete the process of configuring the processor as a gateway. Further, this additional information may include, for example, the number and names for the gateways.

[0118] Once the processor is configured as a gateway, the network operations center **610** may prevent the gateway from connecting to the public web server **611** when exchanging additional information with the network operations center **610**. For example, after a configured gateway contacts the network operations center **610**, the network operations center **610** may reroute any connections to the public web server **611** to the tunneling interface **612**, where a secure tunnel is established for exchanging additional configuration information and code to complete the configuration of the gateway.

[0119] For example, during the user's first session with the public web server **611** of the network operations center **610**, the user may connect to the network operations center using a browser configured with the Secure Sockets Layer protocol (SSL). During this initial contact with the public web server, the network operation center **610** may limit the user's range of permissible functions to basic functions until a secure tunnel is established. In one embodiment, the user may be denied the privilege to change firewall rules, administer partner lists, show tunnel status, show partner list information, delete administrators, and/or define groups of gateways. These denied functions may only be performed through a secure and/or authenticated tunnel to the network operation center **610**.

[0120] **FIG. 8** is an exemplary flow chart depicting the steps for configuring a gateway. The user may provide administration information (step **810**); create an administrator login (step **820**); create a password for the administrator's login (step **830**); provide information describing at least one of the gateways **650-652**, LAN **660**, **661**, Internet **620**, and/or other information necessary to configure a gateway capable of participating in one or more virtual private networks established over the Internet **620** (step **840**); and provide a name for each of the gateways **650-652** (step **850**). The administrator may be a user with the authority to establish one or more virtual private networks over the Internet **620**. The steps of **FIG. 8** may be performed in a secure manner when the user uses one or more of gateways **650-652** to connect to the network operations center **610** and to establish a tunnel with the network operations center **610**.

[0121] To provide administrator information (step **810**), the user may use gateway **652** to connect to the network operations center **610** through the Internet **620**. The user may provide the public web server **611** of the network operations center **610** with sufficient information for registering an administrator including, for example, the administrator's name, log-in, password, e-mail address, pager, and phone number. In the exemplary embodiment of **FIG. 6A**, the public web server **611** may collect and store this information in database server **616**. After the user provides this information (step **810**), the network operations center **610** may create an administrator login (step **820**), providing the user with the capability to configure and administer one or more virtual private networks over the Internet **620**.

[0122] To create passwords (step **830**), the user may select a login name and password for administration of the virtual network, such as a virtual private network for the gateways **650-652**. The user may create a login and password for more than one administrator of the virtual private network to permit other users to login, create, administer, and download a disk image for configuring the virtual private network including the gateways. Furthermore, another user name and password may be created for access to a customer support function at the network operations center **610**.

[0123] In providing information about the gateways **650-652**, LAN **661**, **660**, and/or other information for configuring and administering virtual private networks (step **840**), the user may provide one or more of the following information: the IP address; subnet mask; domain name server address; and gateway IP address for each desired gateway. If a fixed IP address gateway is not used for each gateway **650-652**, the administrator may indicate that a dynamic host control protocol (DHCP) is used. Moreover, the administrator may provide other information including, for example, the media access control (MAC) address for a gateway or a proxy server IP address. For example, the network operations center **610** may perform an auto-discovery process to determine certain information about the administrator's existing network configuration. For example, the network operator center **610** may determine the IP address of a gateway by reading the source and destination address on a packet and determine whether the gateway is accessible behind a firewall by sending test packets to the gateway to see if the packets are rejected by the firewall.

[0124] To name each of the gateways **650-652** (step **850**), the user may select a unique name for each of the gateways **650-652**. Moreover, the user may select a name, such as a domain name for each of the configured virtual private networks. Furthermore, the user may select to use a two level naming hierarchy for each of the gateways **650-652**. For example, a two level naming hierarchy may include, for example, domain_name.gateway_name or customer_name.organization_name.

[0125] Based on the information provided by the user, the network operations center may create and/or assemble program code and information for configuring a processor, such as a computer as a gateway capable of participating in one or more virtual private networks established over the Internet **620**. For example, the network operations center **610** and, in particular, administrative server **615** may generate a disk image that includes the program code and information. The user may select to download the disk image during the initial session(s) with the network operations center **610**. Alternatively, the user may select to download the disk image at a later session. The user may also select to receive the disk image in the form of a diskette; may select to store the disk image at the network operations center **610**; and may permit one or more gateways **650-652** to download the disk image after the user's initial session with the network operations center **610**.

[0126] **FIG. 9** is an exemplary flow chart of the steps performed by network operations center **610** to create code and information (see, also, **FIG. 3** at step **330**) for configuring a gateway. The administrative server **615** in the network operations center **610** may gather the information previously provided by the user (step **910**); create a disk image file (step **920**); encrypt the disk image file (step **930**); and send the disk image to the user (step **940**).

[0127] To gather the information provided by the user (step **910**), the administrative server may retrieve the information previously provided by the user (see, e.g., **FIGS. 7 and 8**) and store the information in the database server **616** of the network operations center **610**. The administrative server **615** may then use this information to create a program code for configuring a computer as a gateway, for example, gateways **650-652**. This program code may be formed into a disk image (step **920**).

[0128] The network operations center **610** may encrypt the disk image (step **930**) to provide privacy. To encrypt the disk image file, the network operations center **610** may use an encryption algorithm, such as DES. The network operations center **610** may send the disk image to one or more of the gateways **650-652** (step **940**). The disk image may be sized to fit on a diskette. If the disk image is provided on a diskette, the user may load the diskette onto a computer (e.g., the first gateway **650**) and reboot the computer. Alternatively, the disk image may be loaded onto a communication device, such as a router, switch, or a bridge, enabling them to participate in one or more virtual private networks established over the Internet. Similarly, the disk image may be loaded onto a wireless device, enabling the wireless device (e.g., a cell phone, personal digital assistant, etc.) to participate in one or more virtual private networks established over the Internet **620**.

[0129] **FIG. 10** is an exemplary flow chart depicting the steps for establishing a tunnel to the network operations center and further configuring one or more gateways. A user installs the disk image (step **1010**) into at least one gateway (e.g., the first gateway **650**) and reboots the processor associated with the gateway (step **1020**). When the processor reboots, the gateway executes the program code in the disk image and may execute any other program code required for operation of the gateway (e.g., operating system and drivers).

[0130] By executing the program code, a routing table in the gateway is initialized to a default state, permitting the gateway to find the Internet **620**. The gateway may be configured with one or more of the following: IP addresses, subnet mask, partner list, domain name server address, and the Internet access device address. The network operations center may also determine a virtual IP address for the gateway. The gateway may then execute a daemon (step **1040**) that may perform the following steps: contact the network operations center **610** and/or the tunnel interface module **612** (step **1050**); open a TCP connection to the tunnel interface module **612**; and initiate IPSec tunnels through the TCP tunnels to the tunnel interface module **612** (step **1060**). The tunnel interface module **612** may authenticate the identity of the gateway (step **1070**); update the tunnel database (step **1080**); and establish a connection from the gateway to the controller module **614** (step **1090**). The controller module **614** may then activate a control path (step **1096**), which the network operations center **610** may use to exchange control information with the gateway.

[0131] As each gateway is configured, it may perform the steps of **FIG. 10** to establish a tunnel with the network operations center **610** and exchange through the tunnel, control information, monitoring information, and additional configuration information, such as the latest partner list.

[0132] In step **1010**, the user of the first gateway **650** may install the disk image, enabling the first gateway **650** to reboot and execute the program code resident on the disk image.

[0133] In step **1020**, the user may reboot the first gateway **650** with the program code. One of ordinary skill in the art would recognize that the reboot may take various forms and may include a total reboot of the gateway or, alternatively, a warm reboot where the gateway loads the disk image without affecting the operation of the gateway. Moreover, one of ordinary skill in the art would also recognize that the disk image may also be loaded on a communication device (e.g., a router, a firewall, a wireless device, and etc.) and/or any other processor. Moreover, the rebooting step **1020** may also include running other software including, for example, an operating system, drivers, program code for IPSec tunnels, and/or software capable of providing the functions of a firewall. RFC-2401, R. Atkinson, The Internet Society (1998), titled "Security Architecture for IP," describes, inter alia, IPSec and is incorporated herein by reference in its entirety.

[0134] In step **1030**, the first gateway **650** may configure its IP addresses for the appropriate subnet mask, domain name server, Internet/Intranet access device, and/or Dynamic Host Configuration Protocol (DHCP) server. Moreover, the first gateway **650** may initialize its internal routing table to a default state.

[0135] The first gateway **650** may start the gateway daemon (step **1040**), which may execute some or all of the program code on the disk image. The gateway daemon may contact the network operations center **610** (including the tunnel interface module **612** step **1050**) using a domain name server or an IP address to resolve the address of the network operations center **610**.

[0136] After initial contact with the network operations center **610** is made, the gateway daemon may open a TCP connection to the tunnel interface module **612**. With a TCP tunnel established, the network operations center **610** may provide the gateway daemon with an IP address, permitting the first gateway **650** to make an internal routing table entry. This routing table entry may permit the first gateway **650** to route, for example, traffic associated with controlling a gateway through the TCP tunnel to the network operations center **610** and tunnel interface module **612**. The first gateway **650** may then communicate directly with the tunnel interface module **612** through the TCP tunnel.

[0137] In step **1070**, the first gateway **650** and the gateway daemon running on the first gateway **650** may begin the process of authentication with the network operations center **610**. For example, an Internet Key Exchange (IKE) may be initiated between the network operations center **610** and the first gateway **650**. This is described in RFC-2409, D. Harkins et al., The Internet Society (1998), titled "Internet Key Exchange," which is incorporated herein by reference in its entirety. A key exchange, such as IKE may be implemented using the Free S/WAN program code available at the Free S/WAN website. Alternatively, a shared secret may be presented for authentication.

[0138] During authentication, the first gateway **650** presents a shared secret to the network operations center **610**. The authentication may include presenting a shared secret to the network operations center. In one embodiment, a gateway presented a virtual IP address that included a shared secret. Alternatively, a public key exchange, such as the one provided by the IKE protocol may also be used to authenticate the first gateway **650** with the network operations center **610** and the tunnel interface module **612**. Furthermore, the shared secret or public key may also be used when a gateway authenticates with another gateway during the establishment of a tunnel between the two gateways.

[0139] Moreover, during the authentication process, the tunnel interface module **612** may verify the authenticity of the first gateway **650** with information previously stored (e.g., the shared secret or public key stored during registration) at the database server **616**. For example, the gateway name, virtual IP address of the gateway, and shared secret may be stored in the database server **616** during the initial registration of the first gateway **650**. When the stored shared secret matches the shared secret presented by the first gateway **650**, the identity or authenticity of the first gateway **650** is established. Alternatively, other authentication techniques and/or public key exchange techniques may be used. Moreover, the authentication system may be eliminated in an environment where authenticity and trust are not a concern. Authentication using MD5 is described in RFC-1828, P. Metzger et al., (1995) titled "IP Authentication using Keyed MD5," which is incorporated herein by reference in its entirety. Accordingly, once the first gateway **650** is authenticated with the network operations center **610**, the first gateway **650** may exchange information with the network operations center **610** in a secure manner through an IPSec tunnel. With the first gateway **650** authenticated, the network operations center **610** may update the tunnel database (step **1080**) stored at database server **616**.

[0140] The first gateway **650** may open a connection, such as a TCP connection to the controller module **614** (step **1090**) using the gateway daemon. The TCP connection to the controller module may go through the TCP tunnel to the controller module **614**. For example, the controller module **614** may permit a connection, such as a control path on a predetermined TCP port. The predetermined TCP port may be the only port accessible through the tunnel interface module **612**. As a result, the gateway daemon may initiate the TCP connection through the TCP tunnel to the tunnel interface module **612**, the switch **680**, and one or more of the firewalls **617** to access the control path at the predetermined TCP port (e.g., port **500**) of the controller module **614**. This TCP connection between the controller module **614** and the gateway daemon may serve as the control path for exchanging control information.

[0141] Before establishing the TCP connection between the first gateway **650** and controller module **614**, the network operations center **610** may perform a tunnel database lookup to ensure that the TCP tunnel is a pending tunnel and not an active tunnel. If the TCP tunnel is an active tunnel, the network operations center **610** may provide an alarm. If the TCP tunnel is listed as pending in the tunnel database, the network operations center **610** may establish the control path between the controller module **614** and the tunnel interface module **612**.

[0142] The network operations center **610** may also implement alarms when predetermined events occur that suggest a possible security concern or risk. The network operations center **610** may generate an alarm when one or more of the following conditions exist: an unauthorized computer attempts to authenticate posing as an established gateway; a tunnel flood attack; a failure to authenticate a gateway; a loss of the control path to a gateway; an internal failure within the network operations center **610** or gateway; an IP address of a gateway changes (i.e., if DHCP is not being used); a MAC address of a gateway's network interface card changes; a spoofing attempt; an attempt to authenticate a non-existent or denied gateway; excessive traffic associated with control or monitoring information; a failed attempt to logon (e.g., multiple tries); performance overruns; and authorization failures.

[0143] When the control path is activated by the controller module **614** of the network operations center **610** (step **1096**), the tunnel interface module **612** may exchange control information with the first gateway **650**. Moreover, the network operation center **610** may communicate one or more of the following information with the first gateway **650** through the control path: the virtual IP address of each gateway on the partner list, the partner list, the network settings, media access control (MAC) addresses, IP addresses (e.g., the DHCP server address, the domain name server address, an Internet access device), a check sum, a shared secret, program code for providing, configuring, and/or controlling a firewall, DHCP server code, and a "cookie." This communication may take place using XML files. An exemplary set of XML files is shown below in Tables 1-6.

[0144] In one embodiment, the network operations center periodically receives through the control path monitoring information from the first gateway **660**, such as the number of active tunnels, up/down times for each tunnel, and ping time between tunnels (i.e., latency). The monitoring information may be exchanged using XML files.

[0145] When the control path is activated (step **1096**), the first gateway **650** may notify each of the other gateways that are listed on its partner list. Although steps **1010-1096** are described above with reference to the first gateway **650**, each of the one or more gateways **650-652** may also perform steps **1010-1096**. For example, the first gateway **650** may notify the second gateway **651** that it seeks to establish a third tunnel. The first gateway **650** and the second gateway **651** may then proceed to establish the third tunnel, after the third tunnel is enabled by the network operations center **610**. Alternatively, the network operations center may enable the third tunnel by authorizing the third tunnel before the first gateway **650** and the second gateway **651** establish the tunnel. Accordingly, the first gateway **650** and the second gateway **651** may exchange information in a private and trusted manner through the established third tunnel that is enabled by the network operations center **610**. The details of establishing the third tunnel are provided below.

[0146] **FIG. 11** illustrates two exemplary partner lists **1110** and **1120**, in accordance with an embodiment of the present invention. Each gateway **650-652** may consent to enabling one or more tunnels with another gateway by providing the network operations center **610** with a list of desired gateways from which it consents to enabling one or more tunnels. The network operations center **610** may determine whether two gateways consent to enabling a tunnel between the two gateways. If so, the network operations center **610** may place each gateway on a partner list of the other gateway. Accordingly, the partner list may reflect the mutual consent of the two gateways to enable one or more tunnels between the two gateways. In the embodiment of **FIG. 11**, the network operations center **610** may generate for the first gateway **650** a partner list that lists the second gateway **651** as a partner. Similarly, the network operations center **610** may generate for the second gateway **651** a partner list that also lists the first gateway **650**. If this is the case, the first gateway **650** and the second gateway **651** may mutually consent to enabling one or more tunnels between the first gateway and the second gateway. As a result, the consent may be mutual in that each gateway consents to enabling one or more tunnels with other gateways. The consents may also be independent in that the first gateway **650** and the second gateway **651** may decide independently of each other.

[0147] The network operations center **610** may determine a partner list for each of the gateways enabled by the network operations center **610** and may store the partner list for each enabled gateway. For example, the network operations center **610** may store a partner list for each gateway in a database within the database server **616**. This database may store each gateway's name with a corresponding partner list that includes each partner's virtual IP address, public portion of the public key, firewall information, and other stored information. As a result, the network operations center **610** may enable a tunnel between the first gateway **650** and the second gateway **651** by determining that each gateway consents to enabling the tunnel and providing sufficient information, such as a partner list that includes each partner's virtual IP address, public portion of the public key, firewall information, etc. to each gateway such that the gateways are capable of establishing the tunnel.

[0148] **FIG. 12** shows an exemplary screen **1250** for adding a gateway to a virtual private network enabled by the network operations center **610**. **FIG. 12** shows that a user may use the screen **1250** to graphically select one or more gateways from which the user's gateway would accept one or more tunnels. The screen **1250** may be presented to the user during the initial configuration of the user's gateway or whenever the user seeks to add a gateway to the user's virtual private network. The network operations center **610** may determine whether a gateway is selected by the user also consents to enabling one or more tunnels to the user's gateway. If the network operations center determines that the selected gateway and the user's gateway mutually consent, the network operations center **610** may place the selected gateway on a partner list for the user's gateway; place the user's gateway on the selected gateway's partner list, and add the selected gateway to the virtual private network depicted in **FIG. 12**.

[0149] **FIG. 13** is an exemplary flow chart depicting steps for establishing a tunnel between at least two gateways in the network **600** shown in **FIG. 6A**. A gateway may seek to establish a tunnel, such as an IPSec tunnel with another gateway that is behind a firewall and is not accessible because the firewall selectively restricts information flowing to the gateway.

[0150] For example, after the first gateway **650** and the second gateway **651** have registered and established control paths with the network operations center **610**, the first gateway **650** may seek to establish a tunnel to the second gateway **651**. The network operations center **610** may enable the tunnel by providing the first gateway **650** with an indication that the second gateway **651** also consents to the enabling the tunnel. The network operations center **610** may acknowledge the mutual consent of the gateways by, for example, placing each gateway on the partner list of the other gateway.

[0151] The network operations center **610** may enable the tunnel by communicating the mutual consent to the first gateway **650** and the second gateway **651**. This consent may be communicated in the form of providing a partner list to each gateway that consents to enabling the tunnel. The partner list may also include configuration information for each gateway listed in the partner list. The configuration information may provide sufficient information for establishing the tunnel and may include, for example, the following for each gateway listed on the partner list: a gateway name, a virtual IP address, a real IP address, and a shared secret for authentication with the network operations center and with other gateways enabled by the network operations center **610**.

[0152] With the partner list, the network operations center **610** may also provide configuration information that includes, for example, firewall information indicating whether a gateway listed on a partner list is accessible or whether the gateway is not accessible behind a firewall. For example, when the first gateway **650** contacts the second gateway **651** (step **1310**) and attempts to establish a tunnel to the second gateway **651** (step **1320**), the first gateway **650** may be notified by the network operation center **610** that the second gateway **651** is behind (i.e., not accessible behind) a firewall. In this example, the network operations center **610** may also provide the first gateway **650** with an indication that the first gateway is behind a firewall.

[0153] If the first gateway **650** is not behind a firewall, the first gateway **650**, as the originating gateway for tunnel request, may determine whether the destination gateway (i.e., the second gateway **651**) is behind a firewall (step **1340**). If the destination gateway (i.e., the second gateway **651**) is not behind a firewall (step **1340**), the first gateway **650** may establish the tunnel to the second gateway **651** (step **1350**) and exchange information with the second gateway **651** through the tunnel (step **1360**). In one embodiment, the gateway with a lower IP address waits for a gateway with a higher IP address to establish a tunnel. In this embodiment, the gateway with the higher IP address is referred to as the originating gateway.

[0154] If the destination gateway (e.g., the first gateway **650**) is not accessible behind a firewall (not shown) (step **1340**), the originating gateway may wait for the destination gateway (e.g., the second gateway **651**) to establish the tunnel (step **1370**). When the second gateway **651** (i.e., the destination gateway) establishes the tunnel, the first gateway **650** and the second gateway **651** may exchange information through the established tunnel (step **1380**).

[0155] If both the originating gateway (e.g., the first gateway **650**) and the destination gateway (e.g., the second gateway **651**) are not accessible behind firewalls (not shown) (steps **1330** and **1390**), a direct tunnel between the originating gateway and the destination gateway may not be possible because the firewall may hide the real or public IP addresses of the originating gateway and destination gateway, respectively. As a result, the network operations center **610** may enable at the proxy module **613** a proxy (also referred to herein as a "Hairpin") (step **1391**) to enable a tunnel between the first gateway and the second gateway **651** through the proxy.

[0156] When the Hairpin is enabled, the originating gateway that is not accessible behind a firewall and the destination gateway that is not accessible behind a firewall may exchange information through the Hairpin, bypassing the firewall of the other gateway (step **1392**). The proxy module **613** may function as a Hairpin that may be enabled by the network operations center **610**.

[0157] In one embodiment, the proxy module **613** may forward packets from one TCP port to another TCP port without examining the contents of the packets (e.g., reading the payload or decrypting the payload). Although the proxy module **613** shown in **FIG. 6A** may reside in the network operations center **610**, the proxy module **613** may reside within any other device in the base network including, for example, another gateway. For example, if two gateways **650**, **651** need a Hairpin, the third gateway **652** may serve as a Hairpin.

[0158] If the originating gateway is accessible a firewall (not shown) (step **1330**) and the destination gateway is not behind a firewall (step **1390**), the originating gateway may open a tunnel to the destination gateway (step **1393**) and proceed to exchange information with destination gateway (step **1395**) through the established tunnel.

[0159] **FIG. 14** depicts a tunnel **1430** established between a first gateway **1410** and a second gateway **1420**, in accordance with the steps depicted in the flow chart shown in **FIG. 13**. To establish the tunnel **1430**, the first gateway **1410** may contact the second gateway **1420** (step **1310**) and attempt to establish the tunnel **1430** to the second gateway **1420** (step **1320**). In the embodiment of **FIG. 14**, the second gateway **1420** appears on the partner list of the first gateway **1410** and the second gateway **1420** may include the first gateway **1410** on its partner list. In this embodiment, neither the first gateway **1410** (i.e., the originating gateway) nor the second gateway **1420** (i.e., the destination gateway) is behind a firewall (steps **1330** and **1340**). The first gateway **1410** may then establish the tunnel to the second gateway **1420** (step **1350**) and proceed to exchange information with the second gateway **1420** through the established tunnel **1430** (step **1360**).

[0160] Although the second gateway **1420** is not shown as being behind a firewall in **FIG. 14**, the second gateway **1420** may alternatively be placed behind a firewall. If the second gateway **1420** is placed behind a firewall (step **1340**) and the second gateway is not accessible behind the firewall, the originating gateway (i.e., the first gateway **1410**) may wait for the destination gateway (i.e., the second gateway **1420**) to establish the tunnel **1430** (step **1370**). While the originating gateway waits for the destination to establish the tunnel, the second gateway **1420** establishes a tunnel to the first gateway **1410** since the first gateway **1410** is accessible because it is not behind a firewall.

[0161] **FIG. 15** illustrates a network **1500** that includes a first gateway **1510**, a second gateway **1530**, a network operations center **610**, a proxy module **1520**, a first tunnel **1532**, a second tunnel **1531**, and a control module **614**. The gateways **1510** and **1530** are each behind firewalls **1590**, **1591**, respectively, that selectively restricts access to each of the gateways **1510**,**1530**. In this embodiment, the proxy module **1520** may reside in the network operations center **610**. The first gateway **1510** may be the originating gateway that is not accessible behind a firewall **1590** (step **1330**). Because the destination gateway (i.e., the second gateway **1530**) may not be accessible behind a firewall **1591** (step **1390**), the first gateway **1510** may not establish a tunnel directly to the second gateway **1530** and instead may use the proxy module **1520** as a Hairpin, bypassing the firewall **1591** of the second gateway **1530**.

[0162] To enable the Hairpin (step **1391**), the first gateway **1510** may use the configuration data provided by the network operations center **610** to determine that the second gateway **1530** is not accessible behind the firewall **1591**. Alternatively, the first gateway may determine that the second gateway **1530** is not accessible behind the firewall **1591** through other means, such as sending packets to a real IP for the second gateway **1530**. The first gateway **1510** may contact the controller module **614** to request enabling a tunnel to the second gateway **1530**. The controller module **614** may then send a message to the proxy module **1520** to enable a Hairpin for the first gateway **1510** and the second gateway **1530**.

[0163] The proxy module **1520** may allocate a TCP port at the proxy module **1520** for the first gateway **1510** and another TCP port for the second gateway **1530**. The proxy module **1520** may then provide the first gateway **1510** with the TCP port information and provide the second gateway **1530** with the other TCP port information. The proxy module **1520** may then initiate a TCP forwarding process that listens to both TCP ports allocated to the first gateway **1510** and the second gateway **1530**, respectively. The con-

troller module **614** may then proceed to inform the first gateway **1510** through the control path to establish a tunnel **1531** to the proxy module **1520** at the IP address of the proxy module **1520** and at the TCP port previously allocated to the first gateway **1510**. The controller module **614** may also inform the second gateway **1530** to establish a separate tunnel **1532** to the proxy module **1520** at the IP address and at the TCP port allocated to the second gateway **1530**.

[0164] The first gateway **1510** may then proceed to open a TCP connection to the TCP port previously allocated to the first gateway **1510** at the proxy module **1520**. Similarly, the second gateway **1530** may open a TCP connection to the TCP port previously allocated to second gateway **1530** at the proxy module **1520**. The proxy module **1520** may use the TCP protocol to forward TCP packets received from the first gateway **1510** to the second gateway **1530** and forward TCP packets received from the second gateway **1530** to the first gateway **1510**. In the embodiment of **FIG. 15**, a tunnel from each of the gateways **1510,1530** to the network operations center **610** may provide out-of-band signaling to enable the Hairpin at the proxy module **1520**.

[0165] Accordingly, the proxy module **1520** may provide the capability to establish a tunnel between the first gateway **1510** and the second gateway **1530** by bypassing their respective firewalls **1590,1591**. Since firewalls may be configured to allow TCP traffic to originate from behind a firewall (i.e. outbound) but not allow arbitrary TCP traffic in (i.e. inbound), the first gateway **1510** and the second gateway **1530** may both send their respective TCP traffic to the proxy module **1520**. Using TCP forwarding, the proxy module **1520** may act as a proxy to enable the exchange of information through a Hairpin even when the originating gateway and the destination gateway are both behind firewalls that selectively restrict access to the originating and destination gateways.

[0166] The network operations center **610** may control a firewall that selectively allows in-bound and out-bound traffic (e.g., firewalls **1590, 1591**) based on a set of rules. For example, the rules may be used to restrict all in-bound and all out-bound traffic through the tunnels **1531, 1532**. Furthermore, the network operations center **610** may turn-off the rules, thus allowing an in-bound and out-bound traffic through the firewall. Although the firewalls shown in **FIG. 15** reside outside of their respective gateways **1510** and **1530**, the firewalls **1590** and **1591** may alternatively reside in their respective gateways **1510** and **1530**.

[0167] If the network operation center **610** allows in-bound and out-bound traffic through the firewalls **1591, 1592** based on a set of rules, the firewalls **1590, 1591** may each be "on" and may filter packets received from the client side of their respective gateways and the tunnel side of their respective gateways. In this mode, by default, outgoing TCP, UDP, and Internet Control Message Protocol (ICMP) traffic originating on the client side may be allowed to reach the tunnel side. Similarly, the associated return packets from the tunnel side may be allowed to reach the client side. Furthermore, ICMP ping, traceroute traffic, and Domain Name Server (DNS) response traffic (i.e., UDP traffic including responses to a DNS request that originates from a processor on the client side) may also be allowed to reach the client side from the tunnel side. Finally, all other traffic originating from any other source on the tunnel side may be blocked.

[0168] The network operations center **610** may prompt the user of the network **1500** to select particular protocols that pass from the tunnel side to the client side. For example, the network operations center **610** may prompt a user of the gateway **1510** to select additional protocols, such as file transfer protocol (FTP), hypertext transfer protocol (HTTP), secure socket layer protocol (SSL), mail retrieval protocols (e.g., POP3), simple mail transfer protocol (SMTP), and remote login protocol (e.g., TELNET). The user may also be prompted to create additional firewall parameters, such as selecting an allowable protocol, port, and direction for packets allowed through a firewall. For example, when a user is prompted to select an allowable protocol, port number, and direction, the user may select a TCP port number at a gateway to serve as a destination port for all TCP/IP packets received from the tunnel side of the firewall.

[0169] In another embodiment, a firewall maybe "on" and all client side and tunnel side packets other than packets destined for a tunnel enabled by the network operations center **610** are blocked.

[0170] The network operations center **610** may also turn-off the rules associated with a firewall. In this mode, the firewall is essentially "off" and packets are allowed to reach the client side of the firewall from the tunnel side.

[0171] **FIG. 16A** shows a network **1600A** that includes a gateway **1610**, a tunnel **1620**, and the network operations center **610**. The network operations center **610** may include a tunnel interface module **1630**, a controller module **640**, a database server **616** with an administrative server **1618**. The gateway **1610** may include a gateway daemon as described above. The gateway **1610** may include a TCP tunnel driver that generates TCP packets forming a TCP tunnel that encapsulates an IPSec tunnel; an IPSec program code, such as the IPSec program code provided by Free S/Wan to establish the IPSec tunnel; and a virtual device adapter that functions as a virtual network interface card for recognizing a virtual IP address corresponding to the gateway **1610**. The tunnel **1620** may include a data path for voice, video, and/or data and a control path for control and monitoring information.

[0172] **FIG. 16B** illustrates a network **1600B** that includes a gateway **1610**, a client **1615**, a tunnel **1620**, the network operations center **610**, and a local area network **1617**. The client **1615**, which may include a processor such as a personal computer or any other processing device, may connect to the gateway **1610** through the local area network **1617**. The gateway **1610** may then route the client's **1615** packets through the tunnel **1620** to a destination, such as the network operations center **610**. Alternatively, the gateway **1610** may route the client's **1615** packets to other gateways (not shown) through one or more tunnels that are enabled by the network operations center **610**.

[0173] The client **1615** may also use a data path within the tunnel **1620** to retrieve administrative information from the administrative server **1618**. Furthermore, a control path may also be established to the controller **640** through the tunnel interface module **1630**. The control path may carry control information, such as out-of-band signaling information for enabling one or more tunnels from the gateway **1610**. The control information may include, for example, a partner list exchanged between the network operations center **610** and the gateway **1610**.

[0174] **FIG. 17** is an exemplary flow chart for a protocol that may be implemented to communicate between the gateway **1610** and the network operation center **610** shown in **FIG. 16A**. The gateway **1610** may connect to the tunnel interface module **1630** in the network operations center (NOC) **610** using a TCP tunnel (step **1710**) and provide to the tunnel interface mode **1630** a virtual IP address and shared secret to authenticate with the network operations center **610**.

[0175] The tunnel interface module **1630** may use the virtual IP address of the gateway **1610** to search and retrieve a shared secret stored within the network operation center **610** (step **1720**). The shared secret may consist of a simple password, a simple bit string, a public key, or an MD5 hash. Alternatively, a public portion of a Public-Private Key pair may be used for authentication. If the shared secret provided by the gateway **1610** is authentic and thus corresponds to the shared secret that is stored for the gateway **1610** (step **1730**), the gateway **1610** may proceed to negotiate a TCP tunnel (step **1750**) with the tunnel interface module **1630**. If the shared secret is not authentic (step **1730**), the tunnel interface module **1630** may disconnect the gateway **1610** (step **1740**) and generate an alarm (step **1745**).

[0176] To initialize the gateway (step **1760**), the gateway **1610** may send to the tunnel interface module **1630** an initiation message that includes a public portion of the Public-Private Key (PPK) pair (i.e., generated with the RSA algorithm) and a name for the gateway **1610** (step **1750**). In one embodiment, program code compliant with RSA signature algorithm, such as RSAsig program code included in the Free S/WAN may be used to generate the public part of the key pair.

[0177] The network operations center **610** may determine whether to accept or reject a tunnel requested by the gateway **1610** by authenticating that gateway based on the shared secret.

[0178] The gateway **1610** may first request to sign-on to the network operations center **610** (step **1770**). The network operations center **610** may then acknowledges the sign-on request. The gateway **1610** may then proceed to sign-on to the network operations center **610** (step **1770**). This permits the gateway **1610** and the network operations center **610** to exchange configuration information (step **1780**) including, for example, a partner list for the gateway **1610**; virtual IP addresses and real IP addresses for the gateway **1610**, network operations center **610**, and any other gateways on the partner list for the gateway **1610**; and/or public key information for authenticating the gateway **1610** with other gateways and the network operations center **610**. In one embodiment, the configuration information is exchanged using XML files. Further, as the configuration of the gateway **1610** changes, the network operations center **610** may broadcast the configuration information to any other gateway listed on the partner list of the gateway **1610**. Although **FIG. 16A** shows one gateway (e.g., the gateway **1610**), a plurality of gateways (not shown) may connect to the network operations center **610** by performing the steps shown in **FIG. 17**.

[0179] Network operations center **610** may provide a means for a client **1615** to establish a connection via a tunnel of the gateway **1610** to the network operations center **610**. Although **FIG. 16B** shows one client **1615**, a plurality of clients (not shown) may be connected to the gateway **1610**. If a plurality of clients are connected to the gateway **1610**, each of the clients may access one or more tunnels to the network operations center **610** through the LAN **1617** and the gateway **1610**. Accordingly, each of these clients may participate in the virtual private network of **FIG. 16B**.

[0180] Table 1 lists exemplary Extensible Markup Language (XML) name value pairs provided by the network operations center **610** for configuring a gateway. For example, a gateway may receive the configuration information for itself and for each gateway on its partner list. Moreover, a gateway may receive this XML information whenever the gateway is connected to the network operations center **610**.

[0181] Referring to Table 1, the network operations center **610** may provide each gateway enabled by the network operations center with one or more of the following: a gateway name, a domain name for the virtual private network, a virtual Internet Protocol (IP) address, and a public IP address visible to the Internet **620**. Moreover, the network operations center **610** may provide information describing one or more of the following: whether a gateway is accessible behind a firewall; a network configuration for a gateway; whether a dynamic host configuration protocol (DHCP) is used at a gateway; IP addresses of the primary and secondary domain name servers associated with a local area network interfacing a gateway; and an IP address of a local IP proxy device providing Internet access to a local area network interfaced to a gateway.

[0182] Table 2 lists exemplary XML name value pairs provided by the network operations center for configuring a media access layer interface (e.g., an Ethernet interface) at a gateway configured by the network operations center **610**. Moreover, the gateway may receive this configuration information for itself and each gateway on its partner list. The network operations center **610** may provide a name for the media access interface, a local IP address for the media access interface, a gateway IP address for the media access layer interface associated with the gateway, a subnet mask for the media access layer interface associated with the gateway, and whether addresses for the media access layer interface are assigned using a DHCP.

TABLE 1

Configuration Information	
<local computer information>	
computer_name	="org5"
domain_name	="bugwheat2"
virtualip_address	="10.0.11.130"
visibleip_address	="208.185.39.2"
firewall_in_place	="no"
network_config	="Inline (i.e., GATEWAY AND IAD)"
dns_from_dhcp	="no"
dns_primary	="10.10.10.2"
dns_secondary	="10.10.10.3"
Proxyip	="208.185.40.2"
</local computer information>	

[0183]

TABLE 2

Local Interface Information	
<local interface information>	
name	="eth0"
mac_layer_address	="00:90:27:EE:02:3B"
local_IP_address	="208.185.39.2"
gateway	="208.185.39.1"
subnet_mask	="255.255.255.0"
dhcp	="none"
</local interface information>	

[0184] Table 3 lists exemplary XML name value pairs provided by the network operations center for a local area network interfacing a gateway. Moreover, the gateway may receive the information for itself and each gateway on its partner list.

[0185] For example, the network operations center **610** may provide a gateway with information describing a local area network, such as the local area networks **661**, **660** interfacing each of the gateways **650**, **651** shown in **FIG. 6A**. The XML name value pairs may include configuration information describing an IP address range for the local area network, describing one or more members of an Access Control List and whether to include a tunnel access privilege for each member of the Access Control List, and specifying a gateway address for a subnet interfacing the local area network.

TABLE 3

Local LAN Information	
<local_LAN_Information><address range>	
startip_address_range	="208.185.49.1"
endip_address_range	="208.185.49.255"
Type	="included"
Gateway	=""
</address range>	
</local_LAN_Information>	

[0186] Table 4 lists exemplary XML name value pairs for cryptographic information provided by the network operations center **610** to a gateway. For example, a gateway may receive the cryptographic information for itself and each gateway on its partner list. The network operations center **610** may provide the cryptographic information to enable an encrypted information flow, such as an encrypted tunnel between the gateway and another gateway or the network operations center **610**. This cryptographic information may include the type of encryption algorithm, format (e.g., standard associated with the algorithm), the key information for the algorithm (e.g., a public key), and other parameters for the encryption algorithm.

TABLE 4

Cryptographic Information	
<cryptographic key>	
Kind	="PublicKey"
Type	="NOC's_Primary_Key"
Format	="RSA"

TABLE 4-continued

Cryptographic Information	
Encryption	="3DES"
Modulus	="0x . . . 01"
modulus_bits	="1024"
public_exp	="0x03"
</cryptographic key>	

[0187] Table 5 lists exemplary XML name value pairs for firewall information provided by the network operations center **610** to a gateway. For example, the gateway may receive the firewall information for itself and each gateway on its partner list. The firewall information may modify and/or configure a firewall and may include rules for the firewall, such as the protocol type permitted to traverse the firewall, a direction for the permitted protocol, allowable source and destination addresses (e.g., IP addresses and port addresses), a flag to enable the rules, a name for each rule, whether to accept packets from another firewall, and a number indicating the order in which rule is executed in a firewall.

[0188] In one embodiment, Tables 1-5 may be stored in the network operations center **610** and indexed according to gateway name and/or virtual IP address of a gateway. Table **6** lists exemplary XML name value pairs for monitoring information received by the network operations center **610**. In one embodiment, a gateway may provide monitoring information about tunnels enabled by the network operations center **610**. This monitoring information may permit the network operations center **610** to monitor the latency and bandwidth associated with a tunnel. For example, every 5 minutes a gateway may send to the network operations center **610** information corresponding to the accumulated number of packets and bytes transmitted at the gateway; the accumulated number of packets received at the gateway; the minimum round-trip time, maximum round-trip time, and 5 minute average round-trip time (i.e., in milliseconds) for packets traveling between the gateway and each gateway on the partner list of the gateway.

TABLE 5

Firewall Information
<firewall rule>
protocol="tcp"
direction="in"
src_ip_mask="$any"
src_port="1024:65535"
dst_ip_mask="$1"
dst_port="21"
action="ACCEPT"
rule_number="1"
</firewall rule>

[0189]

TABLE 6

Monitoring Information	
<bandwidth>	
time_of_day	="1800Z"
interval	="5"

TABLE 6-continued

Monitoring Information	
xmit_packets	="10000"
xmit_bytes	="160000"
rcv_packets	="5"
rcv_bytes	="40"
</bandwidth>	
<latency>	
tod	="1800Z"
interval	="500"
minimum	="50"
maximum	="500"
average	="100"
</latency>	

[0190] FIG. 18 shows a network 1800 including one or more client computers 1824, 1823 connected to a hub 1822 that interfaces a first gateway 1821. The first gateway 1821 may interface the Internet 1840 through an Internet Access Device (IAD) 1820 (see, e.g., IAD1 in FIG. 18). The hub, gateway, and IAD may be in an in-line configuration. The network 1800 may also include one or more client computers 1834, 1833 that are connected to a hub 1832 interfacing a second gateway 1831. The second gateway 1831 may connect to a second IAD 1830 that provides access to the Internet 1840. The network operations center 610 may also interface the Internet 1840. Although the in-line configuration is shown, other configurations of the network 1800 may also be implemented. For example, the hub 1822 may connect directly to the IAD 1820 instead of connecting to the gateway 1821.

[0191] A tunnel may be enabled between the first gateway 1821 and the second gateway 1831 by the network operations center 610. Once established, the tunnel may pass through the IAD 1820, the Internet 1840, and an IAD 1830.

[0192] FIG. 19 is an exemplary flowchart for detecting address changes in the network 1800 shown in FIG. 18. The network operations center 610 may establish a first tunnel (not shown) to the first gateway 1821 and a second tunnel (not shown) to the second gateway 1831. Each of these tunnels may be established through a base network, such as the Internet 1840 and may permit the network operations center 610 to exchange information including, for example, configuration information and/or monitoring information (see, e.g., Tables 1-6 above) with each of the gateways 1821, 1831 (step 1910).

[0193] To detect an address change (step 1920), the network operations center 610 may monitor the status of each gateway 1821, 1831 through the first and second tunnels, respectively. When a real or public address, such as a real or public IP address of gateway 1821 changes, the network operations center 610 may detect the change by determining that the first tunnel between the network operations center and the gateway 1821 is terminated. For example, when an Internet Service Provider (ISP) changes the public IP address associated with the IAD 1820, the network operations center 610 may drop the first tunnel to the first gateway 1821 and detect an address change at the first gateway 1821 (step 1920). The gateway 1821 may then use its new IP address (i.e., the new public IP address associated with the IAD 1820) to reestablish the first tunnel to the network operations center 610 (step 1930) by performing the steps shown in FIG. 17.

[0194] Before reestablishing the first tunnel, the network operations center 610 may first authenticate the gateway 1821 (e.g., using a public key for gateway 1821). Once the first tunnel is reestablished, the network operations center 610 may then store the new IP address associated with the gateway 1821 (step 1940) and inform other gateways as to the new IP address (step 1950).

[0195] When the public IP address (i.e., the real IP address) of the first gateway 1821 changes, the second gateway may 1831 also drop a third tunnel (not shown) between the second gateway 1831 the first gateway 1821. The first gateway 1821 and the second gateway 1831 may then proceed to reestablish the third tunnel after the first gateway 1821 authenticates with the network operations center 610 and provides the public IP address to the network operations center 610. Although FIG. 18 is described in connection with only two gateways, additional gateways (e.g., the gateways 1810-1815) may also be added to a virtual network, such as a virtual private network enabled by the network operations enter 610.

[0196] In the embodiment of FIG. 18, when additional gateways (e.g., the gateways 1810 through 1815) are present and are included in the partner list of the first gateway 1821, the network operations center 610 may notify the additional gateways and/or computer 1862 as to the new public IP address of the first gateway 1821 (step 1950). For example, the network operations center 610 may broadcast the new public IP address to all of the gateways on the partner list of the first gateway 1821.

[0197] FIG. 20 is an exemplary flow chart for resolving IP address conflicts in a local area network interfacing a gateway. One or more client computers 1823, 1824 interfacing the first gateway 1821 may use IP addresses that are local or private and conflict with the local IP addresses of the client computers 1834,1833 interfacing the second gateway 1831. For example, the locally assigned IP address associated with the clients 1823, 1824 of the first gateway 1821 may be identical and thus may conflict with the locally assigned IP addresses associated with the clients 1833, 1834 of the second gateway 1831. This address conflict may be possible because the IP addresses of the client computers 1824, 1823 may be private or local addresses that are routable within the local area network served by the first gateway 1821. Thus, if a client of the first gateway 1821 has the same IP address as a client of the second gateway 1831, information may not be routed between the clients with conflicting addresses. Although detecting such address conflicts may be applicable in various environments, when an extranet is established, a client may be external to an organization and thus may use a local address that is not compatible with the local addresses used on the organization's network, such as the organization's intranet, wide area network, or local area network.

[0198] An address conflict may be detected when the first gateway 1821 establishes a tunnel to the second gateway 1831 (step 2010). For example, the first gateway 1821 may receive an IP address range (see, e.g., Table 3) for the second gateway 1831 and determine that an address conflict exists. When an address conflict exists during the establishment of the tunnel between the first gateway 1821 and the second gateway 1831, the first gateway 1821 may propose a first intermediate address space (step 2020). The second gateway

1831 may propose a second intermediate address space (step 2030). Each gateway 1821,1831 may then negotiate an intermediate address space that does not conflict with the range of local addresses for the clients interfacing the gateway.

[0199] To negotiate the first intermediate address space and the second intermediate address (step 2040), the second gateway 1831 may accept the first intermediate address space proposed by the first gateway 1821 if the second gateway 1831 finds the first intermediate address space acceptable. An address space may be acceptable when the proposed address space does not conflict with the second gateway's 1831 local addresses. If the second gateway 1831 does not find the first intermediate address space acceptable, the second gateway may request from the first gateway 1821 another first intermediate address space.

[0200] If the first gateway 1821 finds the second intermediate address space proposed by the second gateway 1831 acceptable, the first gateway 1821 may accept the second intermediate address space. If the first gateway 1821 does not find the second intermediate address space acceptable, the first gateway 1821 may request another second intermediate address space from the second gateway 1831.

[0201] The first gateway 1821 and the second gateway 1831 may provide the range of addresses in the first intermediate address space and the second intermediate address, respectively, to the network operations center 610 (step 2050). For example, the first gateway 1821 and the second gateway 1831 may send the first and second virtual address intermediate address ranges to the network operations center 610 through the first and second tunnels, respectively.

[0202] To translate the address of a packet based on the first intermediate address space and the second intermediate address space (step 2060), the first gateway 1821 may convert addresses, such as the IP addresses of packets destined for the second gateway 1831 into the first intermediate address space. The second gateway 1831 may then detect the packets addressed in the first intermediate address space. Similarly, the second gateway 1831 may convert the IP addresses of packets destined for the first gateway 1821 into the second intermediate address space. The first gateway 1821 may also detect the packets addressed in the second intermediate address space. Consequently, each gateway may be responsible for determining if a local address conflict exists with another gateway; resolving the address conflict; and translating addresses of the packets to and from the negotiated address space such that the translation is transparent to clients interfacing each gateway.

[0203] As additional gateways are added to the network 1800, each additional gateway may establish one or mores tunnels enabled by the network operations center 610 (step 2010); propose and negotiate an intermediate address space(s) if an address conflict exists with another gateway (steps 2020-2040); send the intermediate address space(s) to the network operations center 610 (step 2050); and translate packets to and from the negotiated intermediate address spaces(s) (step 2060).

[0204] For example, when a third gateway 1810 is added to the network 1800, the third gateway 1810 may establish a tunnel enabled by the network operations center 610 to the first gateway 1821 (step 2010). The third gateway 1810 may also perform the steps 2020-2060 if an IP address conflict exists with the clients 1824,1823 of the first gateway 1821. The third gateway 1810 may then establish a tunnel to the second gateway 1821 and perform steps 2020-2060 if an address conflict exists with the clients 1834, 1833 of the second gateway 1831. As each gateway is added to the network 1800, the added gateway may negotiate an intermediate address space with each existing gateway to resolve any local address conflicts. Accordingly, one or more intermediate address spaces may be negotiated in a pair-wise manner between pairs of gateways enabled by the network operations center 610.

[0205] FIG. 21 is a block diagram of another exemplary virtual private network 2000 enabled by the network operations center 610. The network 2000 may include a first computer 2100, a second computer 2200, a network operations center 610, and a gateway 650 connected to a local area network 660 that includes one or more host or client computers 2662, 2663 and servers 2661, 2664. Moreover, the network 2000 may include one or more tunnels 2300, 2700, 2800 enabled by the network operations center for exchanging information between first computer 2100, second computer 2200, and gateway 650 and one or more tunnels 2400, 2500, and 2600 for exchanging information including configuration information and/or monitoring information (see, e.g., Tables 1-6) with the network operations center 610.

[0206] The host computers 2662, 2663 and servers 2661, 2664 may include computers similar to the host computers 154,155. Furthermore, the servers 2661, 2664 may include servers that support printing, file sharing, electronic mail, image storage, video storage, application hosting, hosting network services, and other functions capable of being hosted on a server.

[0207] The first computer 2100 and the second computer 2200 may include processors, such as the host computers 154 and 155. In one embodiment, the first computer 2100 and the second computer 2200 may include a Windows™ operating system. Alternatively, the first computer 2100 and the second computer 2200 may include a Linux operating system. The first computer 2100 and the second computer 2200 may each be capable of establishing tunnels enabled by the network operations center 610.

[0208] The first computer 2100 and the second computer 2200 may be part of different subnets. If that is case, the network operations center 610 may assign a virtual IP address to the first computer 2100 and another virtual IP address to the second computer 2200 and resolve any local address conflicts using, for example, the steps shown in FIG. 20. Unlike the gateway 650 that routes information to host computers 662, 663 and servers 661, 664, the first computer 2100 and the second computer 2200 are standalone computers that may route packets to a tunnel 2300, 2700, 2800. Moreover, unlike the gateway 650 that may maintain a dedicated control path 2600 to the network operations center 610, the first computer 2100 and second computer 2200 may each connect to the network operations center 610 through tunnels 2400, 2500 when required to exchange control and/or monitoring information with the network operations center 610.

[0209] To enable a tunnel between the first and second computers **2100, 2200**, the network operations center **610** may enable the tunnel **2300** between the first and second computers **2100, 2200** after the first and second computers **2100, 2200** perform the steps shown in **FIG. 17** (see, e.g., steps **1710-1780**). For example, in the embodiment of **FIG. 21**, the first computer **2100** may connect to the network operations **610** through the tunnel **2400** to exchange information, such as Tables **1-6** above. This information may include an indication that the first computer **2100** consents to the establishment of the tunnel **2300** with the second computer **2200**. The second computer **2200** may also connect to the network operations **610** through the tunnel **2500** to exchange information and to indicate consent to enabling the tunnel **2300** between the first computer **2100** and the second computer **2200**.

[0210] After indicating consent and the network operation center **610** enabling the tunnel **2300**, the first computer **2100** and/or the second computer **2200** may disconnect the tunnels **2400, 2500** and establish the enabled tunnel **2300**.

[0211] The first computer **2100** and/or the second computer **2200** may reconnect tunnels **2400, 2500** to the network operations center when necessary to exchange information. For example, if the address of the first computer **2100** changes, the second computer **2200** may drop the tunnel **2300** to the first computer **2100**. The first computer **2100** may reestablish the tunnel **2400**, authenticate with the network operations center **610**, and provide a new IP address for the first computer **2100**. Similarly, the second computer **2200** may reestablish the tunnel **2500**, authenticate with the network operations center **610**, and receive the new IP address for the first computer **2100**. The first computer **2100** and second computer **2200** may then disconnect the tunnels **2400, 2500** to the network operations center **610** and reestablish the tunnel **2300**.

[0212] If the first computer **2100** has limited communications capability, a user of the first computer **2100** may dial in to the network operations center **610** using a wired or wireless Internet connection to create the tunnel **2400**. For example, the first computer **2100** may include a mobile processor, such as a laptop computer, a personal digital assistant, or an Internet appliance or any other processor capable of establishing one or more tunnel enabled by the network operations center **610**. Using the first computer **2100**, the user may exchange over the tunnel **2400** configuration information to enable one or more tunnels. The first computer **2100** may then disconnect the tunnel **2400** to the network operations center **610** and then establish a tunnel **2700** to the gateway **650** to exchange information securely with the host computers **2662, 2663** or servers **2661-2664** interfacing the gateway **650** through the local area network **660**. As a result, the user of the first computer **2100** may exchange information securely in mobile and/or wireless environments.

[0213] In the embodiment of **FIG. 21**, the network operations center **610** may also enable one or more tunnels between networks that are administered independently of each other or are otherwise incompatible with each other, thus enabling instant extranets. For example, if a user seeks to provide limited access through gateway **650** to one or more resources of LAN **660**, such as a server **2661**, the gateway **650** may consent to enabling a tunnel from an external network or processor, such as computer **2100** and/or computer **2200**. In one embodiment, the computers **2100, 2200** may not have addresses, protocols, or security features that are compatible with those of the gateway **650**. Moreover, the gateway **650** may deny the computers **2100, 2200** access to other resources on the LAN **660**, limiting access only to the server **2664** based on an access control list provided by the network operations center **610**.

[0214] The above embodiments and other aspects and principles of the present invention may be implemented in various environments. Such environments and related applications may be specially constructed for performing the various processes and operations of the invention or they may include a general-purpose computer or computing platform selectively activated or reconfigured by program code (also referred to as code) to provide the necessary functionality. The processes disclosed herein are not inherently related to any particular computer or other apparatus, and may be implemented by a suitable combination of hardware, software, and/or firmware. For example, various general-purpose machines may be used with programs written in accordance with teachings of the present invention, or it may be more convenient to construct a specialized apparatus or system to perform the required methods and techniques.

[0215] The present invention also relates to computer readable media that include program instruction or program code for performing various computer-implemented operations based on the methods and processes of the invention. The media and program instructions may be those specially designed and constructed for the purposes of the invention, or they may be of the kind well-known and available to those having skill in the computer software arts. Examples of program instructions include for example micro-code, machine code, such as produced by a compiler, and files containing a high-level code that can be executed by the computer using an interpreter.

[0216] Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and examples be considered as exemplary only, with a true scope and spirit of the invention being indicated by the following claims.

What is claimed is:

1. A method for enabling a network between a first processor and a second processor using at least one additional processor separate from the first processor and the second processor, the method comprising the steps of:

receiving, at the at least one additional processor, information indicating a consent on behalf of the first processor to enabling a tunnel between the first processor and the second processor;

receiving, at the at least one additional processor, information indicating a consent on behalf of the second processor to enabling a tunnel between the second processor and the first processor;

determining a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the network; and

providing, by the at least one additional processor, to each of the first and second processors the first and second virtual addresses to enable one or more tunnels between the first and the second processors.

2. The method of claim 1, further comprising the step of:

establishing, by the first processor, one or more tunnels to the second processor using the first and second virtual addresses.

3. The method of claim 2, wherein said step of establishing further comprises the step of:

establishing each of the one or more tunnels as an encrypted tunnel.

4. The method of claim 3, wherein said step of establishing each of the one or more tunnels further comprises the step of:

establishing each of the one or more tunnels as an encrypted tunnel that is encapsulated within a protocol.

5. The method of claim 3, wherein said step of establishing each of the one or more tunnels further comprises the step of:

establishing each of the one or more tunnels as an encrypted tunnel based on an Internet Protocol Security (IPSec) tunnel.

6. The method of claim 1, wherein said step of receiving, at the at least one additional processor, information indicating a consent on behalf of the first processor, further comprises the step of:

receiving a name that identifies the second processor as consenting to enabling the one or more tunnels between the first and second processors.

7. The method of claim 1, wherein said step of receiving, at the at least one additional processor, information indicating a consent on behalf of the first processor, further comprises the step of:

receiving, at the at least one additional processor, information indicating the consent on behalf of the first processor to enabling one or more other tunnels between the first processor and other processors separate from the at least one additional processor and the second processor.

8. The method of claim 1, wherein said step of determining further comprises the step of:

selecting, at the at least one additional processor, each of the first and second virtual addresses from a predetermined address range.

9. The method of claim 8, wherein said step of selecting further comprises the step of:

defining each of the addresses in the predetermined address range as an address that is routable through the network when the network is enabled by the at least one additional processor.

10. A method for enabling a network between a first processor and a second processor using at least one additional processor separate from the first processor and the second processor, the method comprising the steps of:

establishing a first tunnel between the first processor and the at least one additional processor;

establishing a second tunnel between the second processor and the at least one additional processor;

determining, at the least one additional processor, whether the first and second processors mutually consent to enable a third tunnel between the first and second processors;

determining a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the network; and

providing the first virtual address to the second processor through the second tunnel and the second virtual address to the first processor through the first tunnel after the at least one additional processor determines that the first and second processor mutually consent to enabling the third tunnel.

11. The method of claim 10, wherein said step of determining, at the at least one additional processor, further comprises the step of:

receiving, at the at least one additional processor, a consent from each of the first and second processors independently of each other.

12. The method of claim 10, further comprising the step of:

establishing, by the first processor, the third tunnel to the second processor using the provided first and second virtual addresses.

13. The method of claim 12, wherein said step of establishing the third tunnel, further comprises the step of:

establishing, by the first processor, the third tunnel to the second processor through a firewall associated with the second processor.

14. The method of claim 12, wherein said step of establishing the third tunnel, further comprises the step of:

establishing the third tunnel as an encrypted tunnel.

15. The method of claim 14, wherein said step of establishing the third tunnel as an encrypted tunnel further comprises:

establishing the third tunnel as an encrypted tunnel that is encapsulated within a protocol.

16. The method of claim 14, wherein said step of establishing the third tunnel as an encrypted tunnel further comprises the step of:

establishing the third tunnel as an encrypted tunnel based on an Internet Protocol Security (IPSec) based encrypted tunnel.

17. The method of claim 12, further comprising the step of:

excluding from the first and second tunnels information that flows from the second processor to the first processor through the established third tunnel.

18. The method of claim 12, further comprising the step of:

excluding from the first and second tunnels information that flows from the first processor to the second processor through the established third tunnel.

19. The method of claim 12, further comprising the step of:

interfacing the first processor to one or more other processors separate from the first and second processors

and the at least one additional processor such that information is routed to the one or more other processors from the second processor through the established third tunnel and the first processor.

20. The method of claim 12, further comprising the step of:

interfacing the first processor to one or more other processors separate from the first and second processors and the at least one additional processors such that information is routed to the second processor from the one or more other processors through the first processor and the established third tunnel.

21. The method of claim 12, further comprising the steps of:

providing, by the at least one additional processor, code and information that uniquely identifies the first processor in the network; and

executing the provided code on the first processor to configure, based on the provided information, the first processor as a gateway to one or more other processors separate from the first and second processors and the at least one additional processor; and

routing, at the configured first processor, information from the one or more other processors through the established third tunnel to the second processor.

22. The method of claim 21, wherein the step of providing further comprises the step of:

providing the code and the information on one or more computer readable media.

23. The method of claim 22, wherein the step of providing further comprises the step of:

downloading the code and the information from the at least one additional processor.

24. The method of claim 10, further comprising:

receiving, from the first processor at the at least one additional processor, information indicating a consent on behalf of the first processor to enable the third tunnel to the second processor.

25. The method of claim 24 wherein said step of receiving, from the first processor further comprises the step of:

receiving a name that identifies the second processor as consenting to enabling the third tunnel between the first and second processors.

26. The method of claim 25, wherein said step of receiving the name further comprises the step of:

receiving the name from the first processor through the first tunnel.

27. The method of claim 10, further comprising the step of:

receiving, at the at least one additional processor, information indicating a consent on behalf of the first processor to enable one or more other tunnels to other processors separate from the at least one additional processor and the second processor.

28. The method of claim 10, wherein said step of determining a first virtual address further comprises the step of:

selecting, at the at least one additional processor, each of the first and second virtual addresses from a predetermined address range.

29. The method of claim 28, wherein said step of selecting further comprises the step of:

defining each of the addresses in the predetermined address range as an address that is routable through the network when the network is enabled by the at least one additional processor.

30. The method of claim 10, further comprising the step of:

providing, from the at least one additional processor and through the first tunnel, information to a firewall that selectively restricts a flow of information into the first processor such that information flowing from the second processor on the enabled third tunnel is allowed by the firewall into the first processor.

31. The method of claim 10, further comprising the step of:

placing between the first processor and the second processor a firewall to selectively restrict a flow of information into the first processor.

32. The method of claim 10, wherein said step of establishing the first tunnel further comprises the step of:

establishing the first tunnel through a proxy server placed between the first processor and the at least one additional processor.

33. The method of claim 10, further comprising the step of:

monitoring, at the at least one additional processor, the first and second processors through the first and second tunnels.

34. A system for enabling a network between a first processor and a second processor, wherein the first and second processors are separate from said system, said system:

a tunneling interface that receives information indicating a consent on behalf of the first processor to enabling a tunnel between the first processor and the second processor, and receives information indicating a consent on behalf of the second processor to enabling a tunnel between the second processor and the first processor; and

a controller that determines a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the network, and that provides to each of the first and second processors the first and second virtual addresses to enable one or more tunnels between the first and the second processors.

35. A computer program product for enabling a network between a first processor and a second processor using at least one additional processor separate from the first processor and the second processor, the computer program product comprising code, said code comprising:

code, at the at least one additional processor, that receives information indicating a consent on behalf of the first processor to enabling a tunnel between the first processor and the second processor, and receives information indicating a consent on behalf of the second

processor to enabling a tunnel between the second processor and the first processor;

code that determines a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the network; and

code, at the at least one additional processor, that provides to each of the first and second processors the first and second virtual addresses to enable one or more tunnels between the first and the second processors.

36. A system for enabling a network between a first processor and a second processor, said system comprising:

at least one memory including

code that receives information indicating a consent on behalf of the first processor to enabling a tunnel between the first processor and the second processor and information indicating a consent on behalf of the second processor to enabling a tunnel between the second processor and the first processor,

code that determines a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the network, and

code that provides to each of the first and second processors the first and second virtual addresses to enable one or more tunnels between the first and the second processors; and

at least one processor, separate from the first and second processors, that executes said code.

37. A system for enabling a network between a first processor and a second processor, wherein the first and second processors are separate from said system, said system:

a tunneling interface that establishes a first tunnel between the first processor and the at least one additional processor and establishes a second tunnel between the second processor and the at least one additional processor;

a controller that determines whether the first and second processors mutually consent to enable a third tunnel between the first and second processors, determines that a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the enabled network, and provides the first virtual address to the second processor through the second tunnel and the second virtual address to the first processor through the first tunnel after the controller determines that the first and second processor mutually consent to enabling the third tunnel.

38. A computer program product for enabling a network between a first processor and a second processor using at least one additional processor separate from the first processor and the second processor, the computer program product comprising code, said code comprising:

code that establishes a first tunnel between the first processor and the at least one additional processor and establishes a second tunnel between the second processor and the at least one additional processor;

code, at the least one additional processor, that determines whether the first and second processors mutually consent to enable a third tunnel between the first and second processors;

code that determines a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the network; and

code for providing the first virtual address to the second processor through the second tunnel and the second virtual address to the first processor through the first tunnel after the at least one additional processor determines that the first and second processor mutually consent to enabling the third tunnel.

39. A system for enabling a network between a first processor and a second processor, said system comprising:

at least one memory including code comprising

code that establishes a first tunnel between the first processor and the at least one additional processor and establishes a second tunnel between the second processor and the at least one additional processor,

code, at the least one additional processor, that determines whether the first and second processors mutually consent to enable a third tunnel between the first and second processor,

code that determines a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the network, and

code that provides the first virtual address to the second processor through the second tunnel and the second virtual address to the first processor through the first tunnel after the at least one additional processor determines that the first and second processor mutually consent to enabling the third tunnel; and

at least one processor, separate from the first and second processors, that executes said code.

40. A network comprising:

a first processor;

a second processor; and

at least one additional processor, separate from the first and second processors, wherein the at least one additional processor determines a first virtual address for the first processor and a second virtual address for the second processor such that one or more tunnels are enabled when the at least one additional processor determines that the first and second processors mutually consent to enabling the one or more tunnels between the first processor and the second processor and provides the second virtual address to the first

processor and the first virtual address to the second processor to enable the one or more tunnels.

41. The network of claim 40 further comprising:

a third processor, separate from the at least one additional processor and the second processor and placed between the first processor and the second processor such that the third processor selectively restricts into the first processor a flow of information on the enabled one or more tunnels.

42. The network of claim 40 further comprising:

one or more other processors, separate from the first and second processors and the at least one additional processors, that interface to the first processor such that information is routed to the second processor from the one or more other processors through the first processor.

43. A system for enabling a network between a first processor and a second processor using at least one additional processor separate from the first processor and the second processor, the system comprising the steps of:

means for receiving, at the at least one additional processor, information indicating a consent on behalf of the first processor to enabling a tunnel between the first processor and the second processor;

means for receiving, at the at least one additional processor, information indicating a consent on behalf of the second processor to enabling a tunnel between the second processor and the first processor;

means for determining a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the network; and

means for providing, by the at least one additional processor, to each of the first and second processors the first and second virtual addresses to enable one or more tunnels between the first and the second processors.

44. A system for enabling a network between a first processor and a second processor using at least one additional processor separate from the first processor and the second processor, the system comprising the steps of:

means for establishing a first tunnel between the first processor and the at least one additional processor;

means for establishing a second tunnel between the second processor and the at least one additional processor;

means for determining, at the least one additional processor, whether the first and second processors mutually consent to enable a third tunnel between the first and second processors;

means for determining a first virtual address for the first processor and a second virtual address for the second processor such that the first and second virtual addresses uniquely identify the first and second processors, respectively, and are routable through the network; and

means for providing the first virtual address to the second processor through the second tunnel and the second virtual address to the first processor through the first tunnel after the at least one additional processor determines that the first and second processor mutually consent to enabling the third tunnel.

* * * * *

US 20010049744A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2001/0049744 A1

Hussey et al. (43) Pub. Date: Dec. 6, 2001

(54) HIGH-SPEED DATA PROCESSING USING INTERNAL PROCESSOR MEMORY SPACE

(76) Inventors: Terrence Hussey, Merrimack, NH (US); Donald W. Monroe, Carlisle, MA (US); Arnold N. Sodder, Newtonville, MA (US)

Correspondence Address:
TESTA, HURWITZ & THIBEAULT, LLP
HIGH STREET TOWER
125 HIGH STREET
BOSTON, MA 02110 (US)

(21) Appl. No.: 09/798,820

(22) Filed: Mar. 2, 2001

Related U.S. Application Data

(63) Non-provisional of provisional application No. 60/186,782, filed on Mar. 3, 2000.

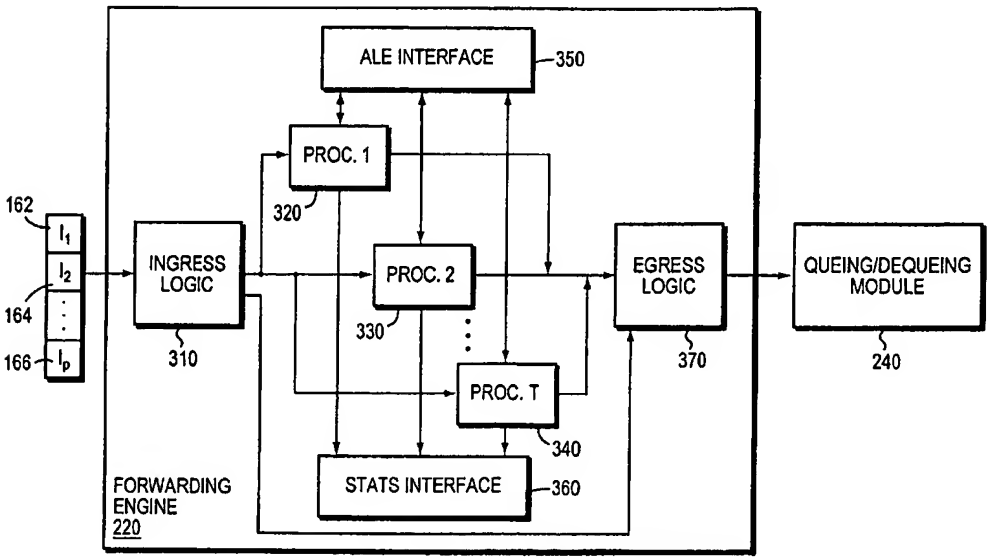
Publication Classification

(51) Int. Cl.[7] G06F 15/173; G06F 15/16

(52) U.S. Cl. 709/238; 709/246

(57) **ABSTRACT**

Significant performance improvements can be realized in data processing systems by confining the operation of a processor within its internal register file so as to reduce the instruction count executed by the processor. Data, which is sufficiently small enough to fit within the internal register file, can be transferred into the internal register file, and execution results can be removed therefrom, using direct memory accesses that are independent of the processor, thus enabling the processor to avoid execution of load and store instructions to manipulate externally stored data. Further, the data and execution results of the processing activity are also accessed and manipulated by the processor entirely within the internal register file. The reduction in instruction count, coupled with the standardization of multiple processors and their instruction sets, enables the realization of a highly scaleable, high-performing symmetrical multi-processing system at manageable complexity and cost levels.



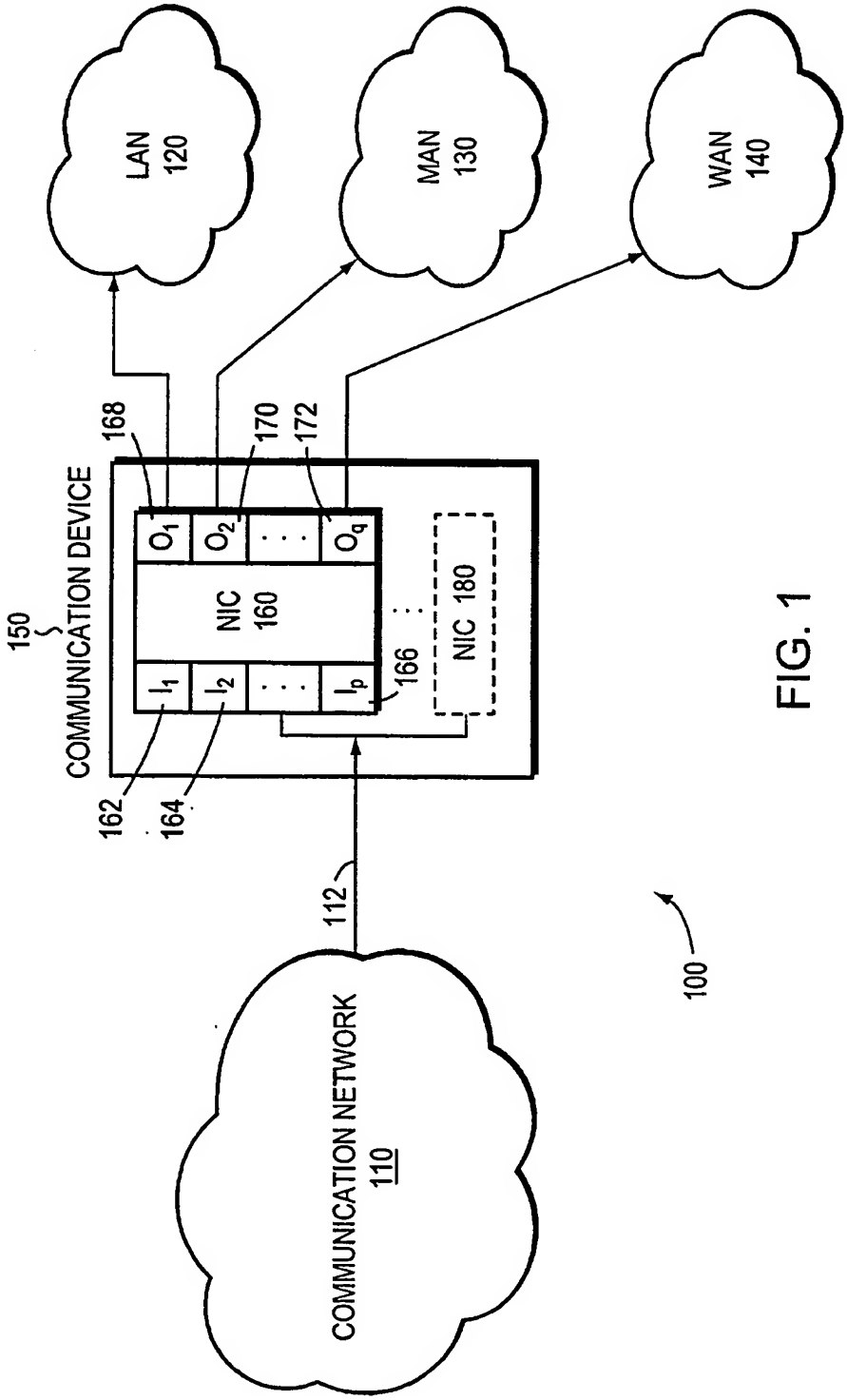

LAN
120
MAN
130
WAN
140
150
COMMUNICATION DEVICE
168
170
172
O_1
O_2
O_q
NIC
160
NIC 180
166
I_1
I_2
I_p
162
164
112
COMMUNICATION NETWORK
110
100

FIG. 1

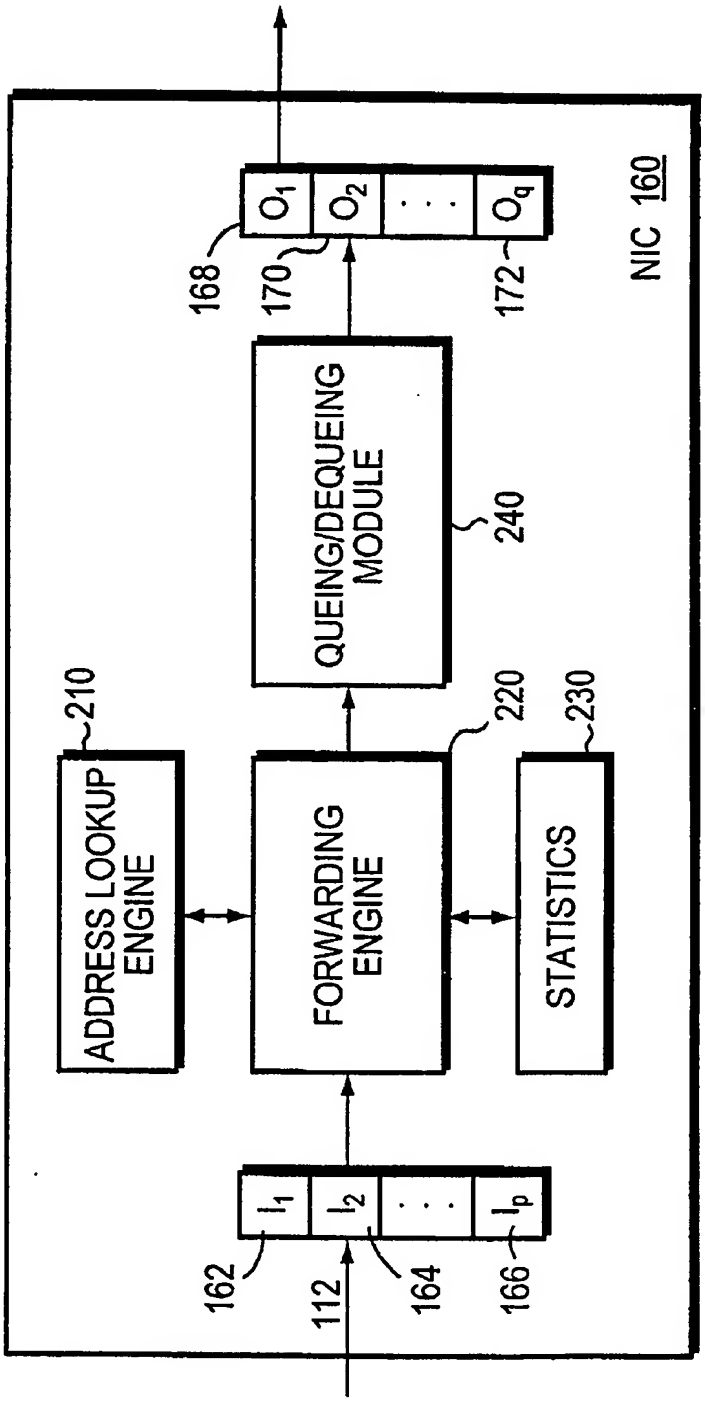

NIC 160
O_1
O_2
O_q
168
170
172
QUEING/DEQUEING MODULE
240
210
220
230
ADDRESS LOOKUP ENGINE
FORWARDING ENGINE
STATISTICS
I_1
I_2
I_p
162
112
164
166

FIG. 2

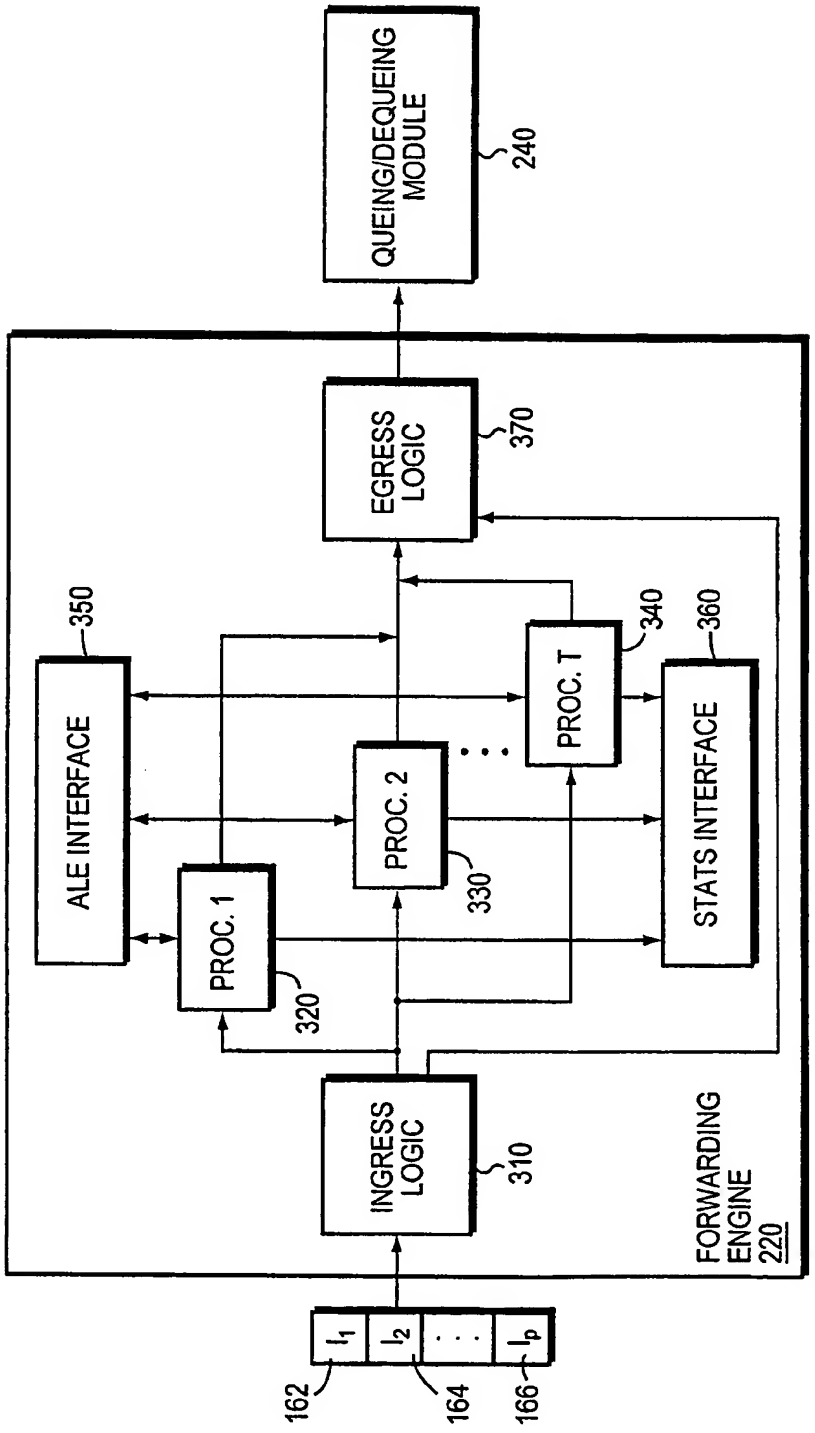

QUEING/DEQUEING MODULE
240
EGRESS LOGIC
370
ALE INTERFACE
350
PROC. T
340
STATS INTERFACE
360
PROC. 2
330
PROC. 1
320
INGRESS LOGIC
310
FORWARDING ENGINE
220
I_1
I_2
I_p
162
164
166

FIG. 3

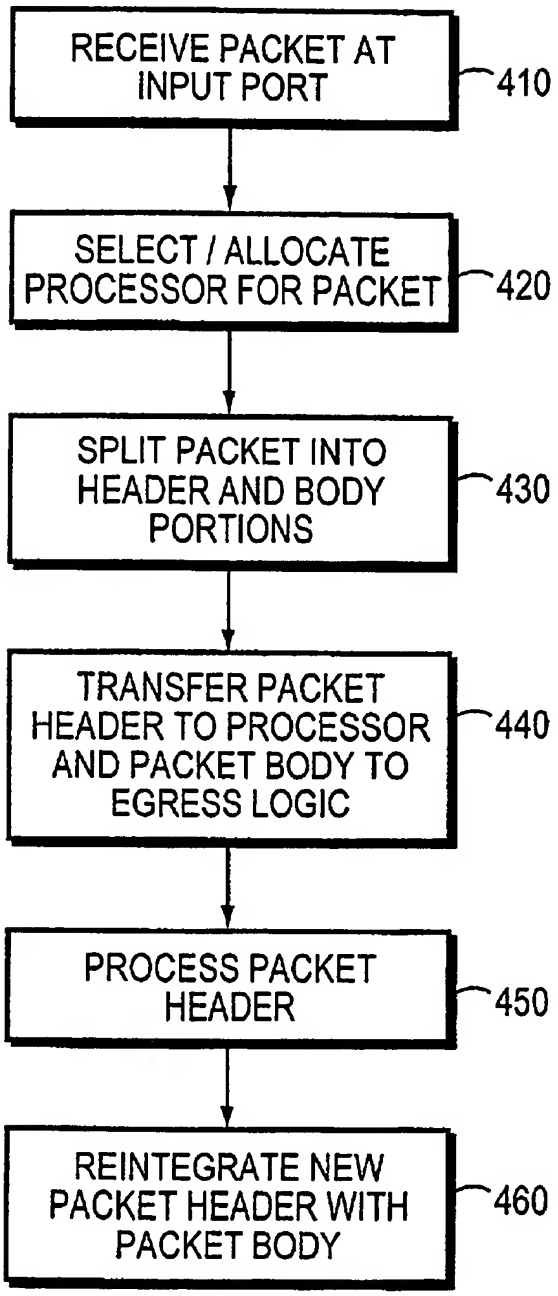

RECEIVE PACKET AT INPUT PORT
410
SELECT / ALLOCATE PROCESSOR FOR PACKET
420
SPLIT PACKET INTO HEADER AND BODY PORTIONS
430
TRANSFER PACKET HEADER TO PROCESSOR AND PACKET BODY TO EGRESS LOGIC
440
PROCESS PACKET HEADER
450
REINTEGRATE NEW PACKET HEADER WITH PACKET BODY
460

FIG. 4

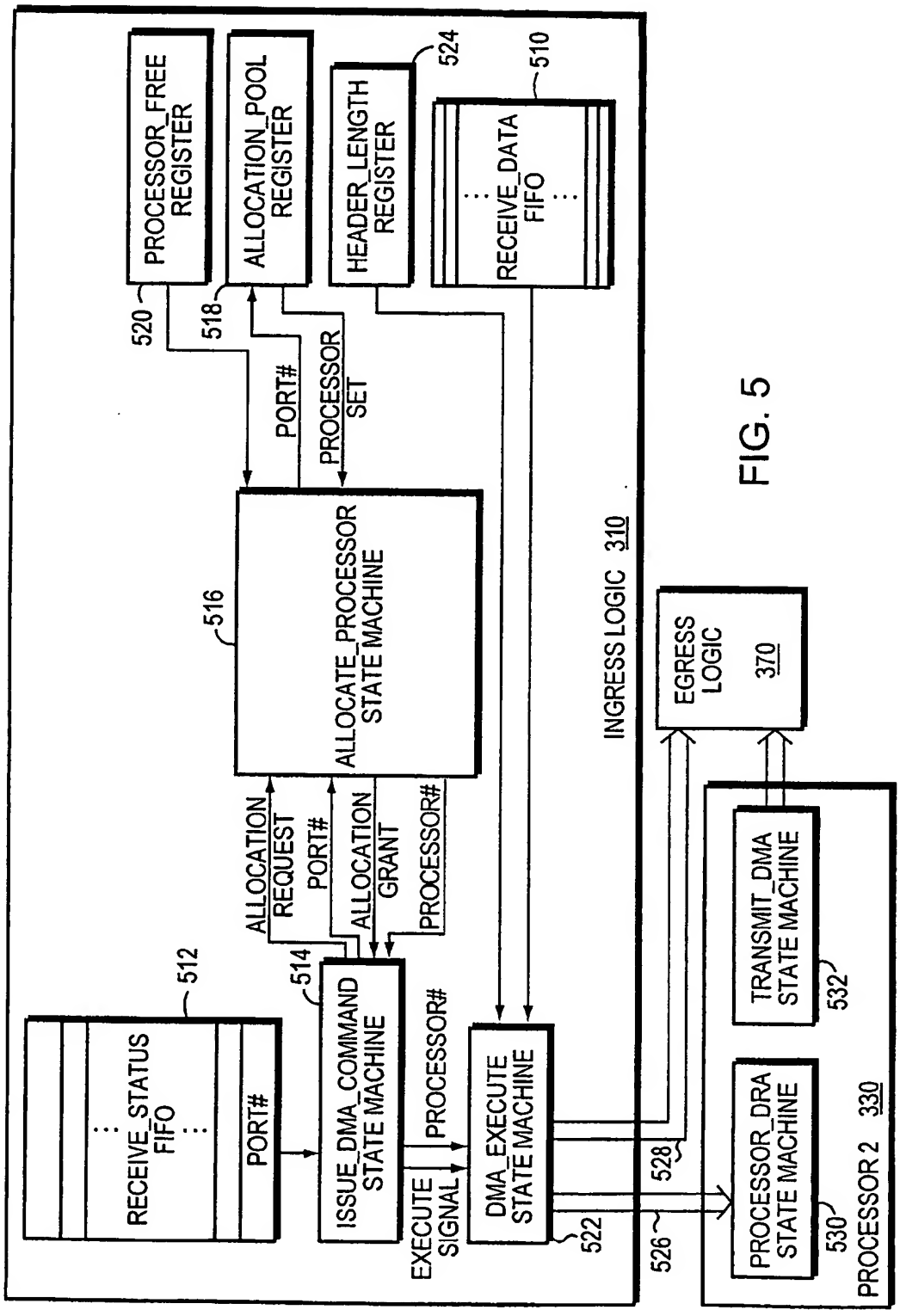

PROCESSOR_FREE REGISTER
ALLOCATION_POOL REGISTER
HEADER_LENGTH REGISTER
524
510
RECEIVE_DATA FIFO
520
518
PORT#
PROCESSOR SET
516
ALLOCATE_PROCESSOR STATE MACHINE
INGRESS LOGIC 310
EGRESS LOGIC 370
ALLOCATION REQUEST
PORT#
ALLOCATION GRANT
PROCESSOR#
512
514
RECEIVE_STATUS FIFO
PORT#
ISSUE_DMA_COMMAND STATE MACHINE
PROCESSOR#
EXECUTE SIGNAL
DMA_EXECUTE STATE MACHINE
522
526
528
TRANSMIT_DMA STATE MACHINE
532
PROCESSOR_DRA STATE MACHINE
530
PROCESSOR 2 330

FIG. 5

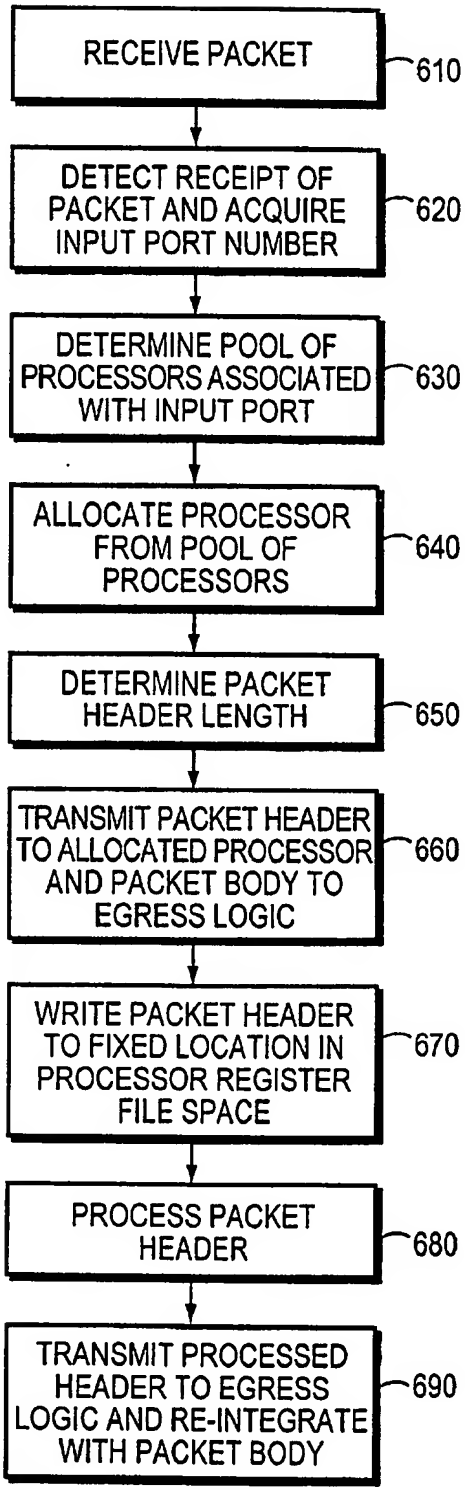

RECEIVE PACKET
610
DETECT RECEIPT OF PACKET AND ACQUIRE INPUT PORT NUMBER
620
DETERMINE POOL OF PROCESSORS ASSOCIATED WITH INPUT PORT
630
ALLOCATE PROCESSOR FROM POOL OF PROCESSORS
640
DETERMINE PACKET HEADER LENGTH
650
TRANSMIT PACKET HEADER TO ALLOCATED PROCESSOR AND PACKET BODY TO EGRESS LOGIC
660
WRITE PACKET HEADER TO FIXED LOCATION IN PROCESSOR REGISTER FILE SPACE
670
PROCESS PACKET HEADER
680
TRANSMIT PROCESSED HEADER TO EGRESS LOGIC AND RE-INTEGRATE WITH PACKET BODY
690

FIG. 6

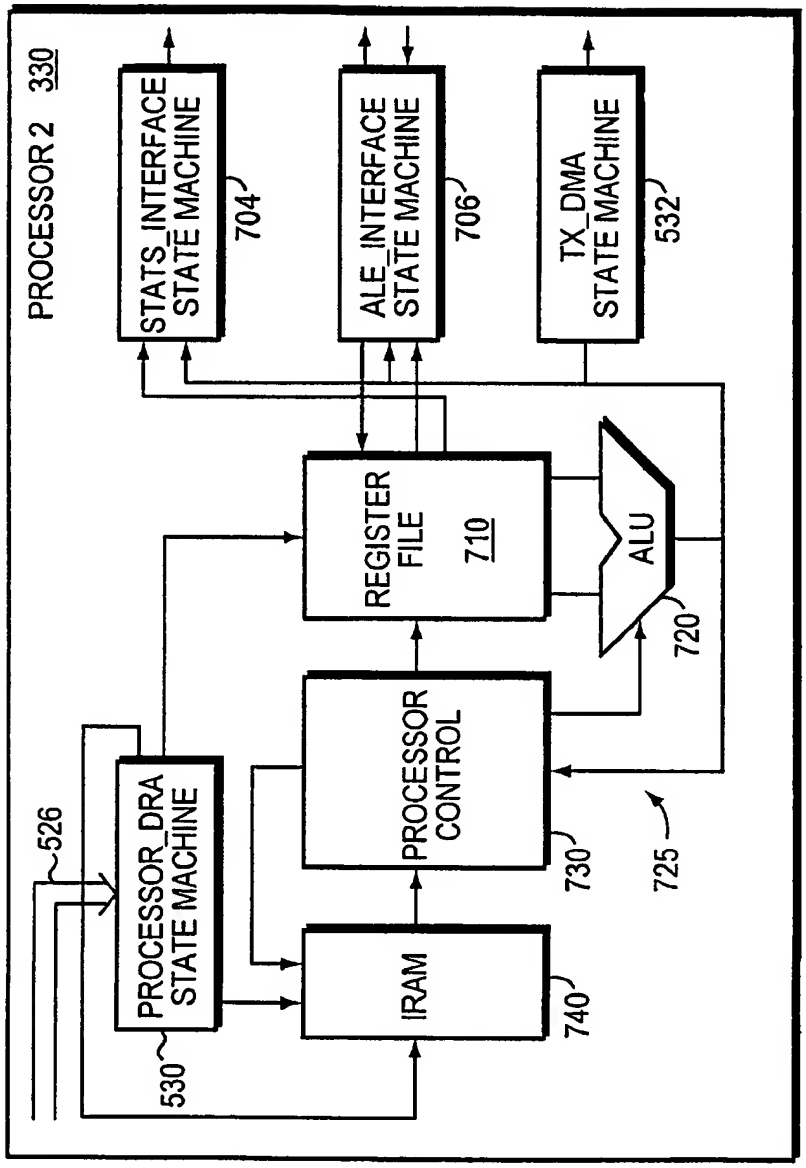

PROCESSOR 2 330
STATS_INTERFACE STATE MACHINE
704
ALE_INTERFACE STATE MACHINE
706
TX_DMA STATE MACHINE
532
REGISTER FILE 710
ALU
720
PROCESSOR CONTROL
730
725
526
PROCESSOR_DRA STATE MACHINE
530
IRAM
740

FIG. 7

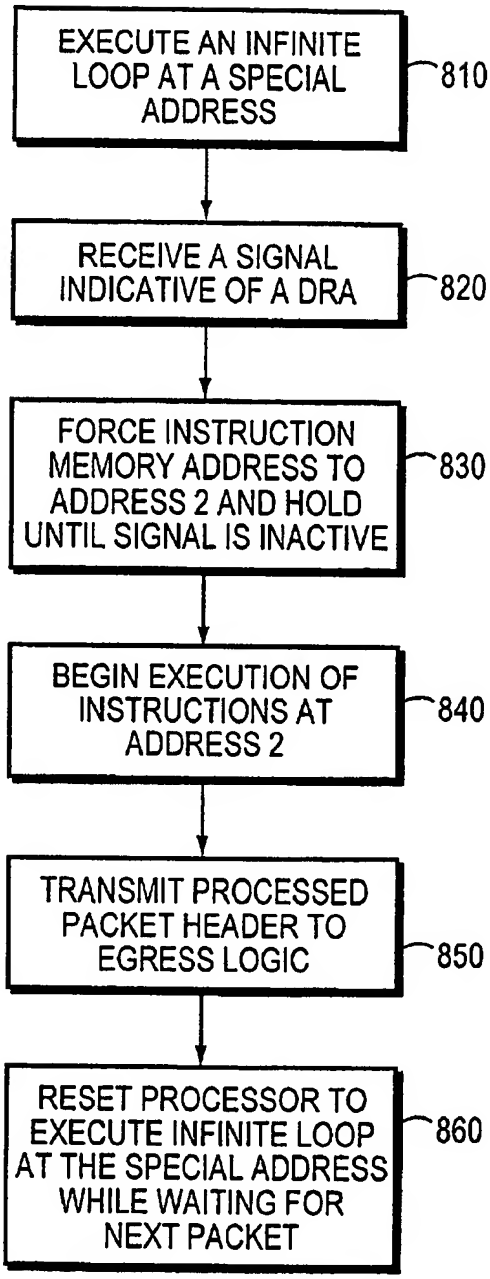

EXECUTE AN INFINITE LOOP AT A SPECIAL ADDRESS
810
RECEIVE A SIGNAL INDICATIVE OF A DRA
820
FORCE INSTRUCTION MEMORY ADDRESS TO ADDRESS 2 AND HOLD UNTIL SIGNAL IS INACTIVE
830
BEGIN EXECUTION OF INSTRUCTIONS AT ADDRESS 2
840
TRANSMIT PROCESSED PACKET HEADER TO EGRESS LOGIC
850
RESET PROCESSOR TO EXECUTE INFINITE LOOP AT THE SPECIAL ADDRESS WHILE WAITING FOR NEXT PACKET
860

FIG. 8

HIGH-SPEED DATA PROCESSING USING INTERNAL PROCESSOR MEMORY SPACE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This claims priority to and the benefit of U.S. provisional patent application number 60/186,782, filed Mar. 3, 2000, the entirety of which is incorporated herein by reference.

FIELD OF THE INVENTION

[0002] The present invention relates generally to information processing, and in particular to the processing activity occurring within internal elements of processors.

BACKGROUND OF THE INVENTION

[0003] Data processing typically involves retrieving data from a memory, processing the data, and storing the results of the processing activity back into memory. The hardware architecture supporting this data processing activity generally controls the flow of information and control among individual hardware units of an information processing system. One such hardware unit is a processor or processing engine, which contains arithmetic and logic processing circuits, general and special purpose registers, processor control or sequencing logic, and data paths interconnecting these elements. In some implementations, the processor may be configured as a stand-alone central processing unit (CPU) implemented as a custom-designed integrated circuit or implemented in an application specific integrated circuit (ASIC). The processor has internal registers for use with operations that are defined by a set of instructions. The instructions are typically stored in an instruction memory and specify a set of hardware functions that are available on the processor.

[0004] When implementing these functions, the processor generally retrieves "transient" data from a memory that is external to the processor, sequentially or randomly loads portions of the data into its internal registers by executing "load" instructions, processes the data in accordance with the instructions, and then stores the processed data back into the external memory using "store" instructions. In addition to loading the transient data into and removing the execution results out of the internal registers, load and store instructions are also frequently used during the actual processing of the transient data in order to access additional information required to complete the processing activity (e.g., accessing status and command registers). Frequent load/store accesses to an external memory is generally inefficient because the execution capability of a processor is substantially faster than its external interface capability. Consequently, the processor often idles while waiting for the accessed data to be loaded into its internal register file.

[0005] This inefficiency can be particularly limiting in devices that operate within communication systems, since the net effect is to constrain the overall data handling capacity of a device and, unless some data is to be dropped rather than transmitted, the maximum data rate of the network itself.

SUMMARY OF THE INVENTION

[0006] The present invention recognizes that frequent accesses to external memory are not necessary for processing a data set that is small enough to be contained within the local register file space of a processor assigned to process the data set. Accordingly, the present invention incorporates data access techniques that are performed, at least in part, independently of the processor and which avoid execution of load and store instructions by the processor.

[0007] In one embodiment, an information processing system and method, incorporating aspects of the present invention, confines the operations of a processor assigned to process a data set within the processor's internal register file. The information processing system comprises a processor, an ingress element, and an egress element. The ingress element receives unprocessed data from an interface to a data source corresponding, for example, to a network interface receiving data from a communications network. The ingress element delivers the unprocessed data, or portions thereof, to the internal register file space of the processor by directly accessing the internal register file space. A unit for manipulating data within the processor (e.g., an arithmetic logic unit) manipulates and processes the data in response to the transfer of the data to the processor's register file and confines its operations entirely within its internal register file space. Upon completion of the processing activity, the egress element directly accesses and removes the processed data from the internal register file space. Alternatively, an intermediate state machine directly accesses the processed data and transfers it to the egress element.

[0008] In one aspect of the invention, one or more state machines are contained within and govern the operation of the ingress and egress elements. One or more state machines are also contained within the processor. The state machines directly access the processor's internal register file space in order to deliver data thereto or remove data therefrom. In one embodiment, the data transfer activities of the state machines are initiated in response to a) receipt of the unprocessed data at the ingress element, b) a signal by processor logic indicating the transfer of unprocessed data into the register file space of the processor, and/or c) a change in the value stored in a logic element, such as a command register.

[0009] The benefits of the present invention can be realized in a number of information processing systems, such as those focused on image processing, signal processing, video processing, and network packet processing. As an illustration, the present invention can be embodied within a communication device, such as a router, to implement network services such as route processing, path determination, and path switching functions. The route processing function determines the type of routing needed for a packet, whereas the path switching function allows a router to accept a packet on one interface and forward it on a second interface. The path determination function selects the most appropriate interface for forwarding the packet.

[0010] The path switching function of the communication device can be implemented within one or more forwarding engine ASICs, incorporating aspects of the present invention, to support the transfer of packets between a plurality of interfaces of the communication device. In this illustrative embodiment, packet data is received by ingress logic associated with a particular input port of a network interface of the communication device via a communications network. A

processor is then selected by the ingress logic from a pool of candidate processors associated with the receive port to process the packet.

[0011] Once the processor has been allocated, the packet is split into header and body portions. The packet header is written into a fixed location within a memory element, such as the internal register file associated with the allocated processor, by at least one state machine of the ingress logic that is configured to write the packet header using direct memory/register accesses and without the processor invoking load or store instructions. The packet body is written to an output buffer. The processor then processes the packet header according to locally stored instructions (again, without invoking load or store instructions) and transfers the processed packet header to a selected output buffer where it is integrated with the packet body and subsequently transferred to a destination output port for transmission from the communication device.

[0012] Prior to receiving the packet header, the allocated processor repetitively executes an instruction stored at a first known location/address in the processor's instruction memory (e.g., address **0**) in an infinite loop. Hardware in the processor detects address **0** to be a "special" address for which hard-wired instructions are returned, rather than instructions from the instruction memory coupled to the processor. When a packet header is transferred to the processor from the ingress logic, a control signal indicates to the processor that the header transfer is in progress. While this signal is active, the processor hardware forces the processor program counter to a nonspecial address (e.g., address **2**), which terminates execution of the infinite loop. Upon completing the transfer of the packet header, the processor begins executing instructions beginning at address **2** of its instruction memory. Once the packet processing activity is complete, the processor is reset (e.g., sets the program counter to address **0**) to repetitively execute instructions at the special address discussed above.

[0013] In this manner, the packet header is directly written to the register file of the processor, without requiring any interaction or prior knowledge by the processor until it is ready to process the packet header. Other information relating to the status or characteristics of the packet (e.g., length) can also be stored locally in the register file using a similar procedure so that the processor need not access an external source to obtain this information.

[0014] To simplify the programming model for multiple processors, a single processor can be allocated for each packet with each of the processors configured to execute a common series of instructions within their respective instruction memories. Enough processors are assigned to ensure that the packets can be processed at the wire/line rate (i.e. maximum bit-rate of the network interface) of the communications network. The reduced instruction set realized when incorporating aspects of the present invention in a plurality of processors in an ASIC reduces the die size of the ASIC, thus enabling a greater density in the number of processors in the ASIC without encountering technological barriers and adverse yield results in the manufacturing of such an ASIC. The ASIC implementation of the present invention is further scaleable, for example, by increasing the clock rate of the processors, by adding more processors to the ASIC, and by aggregating pools of processors (with common instruction sets) from multiple ASICs.

[0015] In one embodiment, the present invention can be used in a symmetric multi-processing (SMP) system, exhibiting a reduced instruction set computer (RISC) architecture, to process packets received over a communications network. The SMP system comprises a plurality of identical processors with common software operating as a pool, any of which is eligible to process a particular packet. Each incoming packet is assigned to an available processor in the pool, and the processors process the packets in parallel using a common instruction set. The SMP system reconstructs the processed packet stream so that it exhibits the proper packet order.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The foregoing discussion will be understood more readily from the following detailed description of the invention, when taken in conjunction with the accompanying drawings, in which:

[0017] **FIG. 1** schematically illustrates a communication device coupling a communication network to other networks, such as LANs, MANs, and WANs;

[0018] **FIG. 2** schematically illustrates several components of a network interface card installed within the communication device of **FIG. 1**, in accordance with an embodiment of the present invention;

[0019] **FIG. 3** schematically illustrates several components of a forwarding engine, which form a portion of the network interface card of **FIG. 2**, in accordance with an embodiment of the present invention;

[0020] **FIG. 4** provides a flow diagram of the steps performed when operating the forwarding engine of **FIG. 3**, in accordance with an embodiment of the present invention;

[0021] **FIG. 5** schematically illustrates several components of the ingress logic and processor of the forwarding engine of **FIG. 3** that perform direct memory and direct register accesses, in accordance with an embodiment of the present invention;

[0022] **FIG. 6** provides a flow diagram of the steps performed during the operation of the ingress logic and processor of **FIG. 5**, in accordance with an embodiment of the present invention;

[0023] **FIG. 7** schematically illustrates a more detailed set of components that form the processor of **FIG. 5**, in accordance with an embodiment of the present invention; and

[0024] **FIG. 8** provides a flow diagram of the steps performed when operating the processor components depicted in **FIG. 7**, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0025] Typical microprocessors execute load and store instructions to load temporary images of data that represent data structures stored in memory elements external to the processor into the processor's local register file for further execution. As used herein, the term "local register file" means the totality of registers within the internal structure of the processor that are available for use in manipulating data. A "register" refers to a distinct group of storage elements,

such as D flip-flops. Depending on processor design, the register file space can be composed of a combination of memory and flip-flops. In any event, the register file is typically implemented using high-speed memory components that provide multiple read and write ports which are independently accessible. During execution of a software program, the typical processor executes a relatively large number of load/store instructions to move data from external memory to the local register file and to move execution results from the local register file to external memory. These frequent accesses to external memory are necessitated because the data set to be processed is too large to fit into the local register file's execution space.

[0026] The present invention recognizes that frequent accesses to external memory are not necessary for processing data sets that are small enough (e.g., 128 to 512, 8-bit data elements) to be positioned entirely within the local register file space. As described in detail below, the present invention incorporates direct memory access (DMA) and direct register access (DRA) techniques to position data and execution results into and out of a processor's register file without the need for the processor to execute instructions, such as load and store instructions, to move the data. In this context, DMA refers to a method which uses one or more state machines to move a block of data into and out of internal or external memory independently of the processor. Similarly, DRA refers to a particular type of DMA, namely, one involving movement of one or more blocks of data into and out of the processor's register file space independently of the processor. In one embodiment, a region of the register file is allocated as a five-port register file space with two write ports and three read ports (as opposed to the normal three-port register file space with one write and two read ports) in order to facilitate direct register file accesses. This approach avoids accesses to external memory that are relatively slow (compared to operations within the register file), avoids memory wait states, and reduces the size of the processor's instruction set. Consequently and in addition to significantly increasing the performance of an individual processor, the die size and power consumption of an application specific integrated circuit (ASIC) containing such processors can be reduced and the overall number of processors in the ASIC can be significantly increased without incurring unsustainable costs.

[0027] Although the present invention will hereafter be described as being implemented in a network interface card of a communication device for the purpose of processing packets received over a network, this particular implementation is merely an illustrative embodiment and those skilled in the art will recognize any number of other embodiments and applications that can benefit from the claimed invention. For example and without limitation, the present invention can benefit information-processing applications involving relatively small data sets, such as those present in image processing, signal processing, and video processing. The present invention can also be implemented in a wide variety of network communication devices (e.g., switches and routers) and other information-processing environments.

[0028] With reference to **FIG. 1**, a communication device **150** receives information (e.g., in the form of packets/frames, cells, or TDM frames) from a communication network **110** via a communication link **112** and transfers the received information to a different communication network or branch such as a Local Area Network (LAN) **120**, Metropolitan Area Network (MAN) **130**, or Wide Area Network (WAN) **140** or to a locally attached end station (not shown). The communication device **150** can contain a number of network interface cards (NICs), such as NIC **160** and NIC **180**, each having a series of input ports (e.g., **162**, **164**, and **166**) and output ports (e.g., **168**, **170**, and **172**). Input ports **162**, **164**, and **166** receive information from the communication network **110** and transfer them to a number of packet processing engines (not shown) that process the packets and prepare them for transmission at one of the output ports **168**, **170**, and **172**, which correspond to a communication network such as the LAN **120**, MAN **130**, or WAN **140** containing the end station.

[0029] With reference to **FIG. 2**, the network interface card (NIC) **160** embodying aspects of the present invention includes input ports **162**, **164**, **166**, a packet processing or forwarding engine **220**, an address lookup engine (ALE) **210**, a statistics module **230**, a queuing/dequeuing module **240**, and output ports **168**, **170**, **172**. The NIC **160** receives packets from the packet-based communication network **110** (**FIG. 1**) at input ports **162**, **164**, **166**. The forwarding engine **220**, together with the ALE **210**, determine the destination output ports of the packets by looking up the appropriate output ports **168**, **170**, **172** associated with that destination, and prepending forwarding vectors onto the packets to aid in routing them to the appropriate output ports.

[0030] The modified packets are delivered to the queuing/dequeuing module **240** where the forwarding vectors are used to organize the packets into queues associated with a particular destination output port **168**, **170**, **172**. The forwarding vectors of each packet are then removed and the packets are scheduled for transmission to the selected output ports **168**, **170**, **172**. The packets are subsequently transmitted from the selected output ports **168**, **170**, **172** to a communication network such as the LAN **120**, MAN **130**, or WAN **140**. In one embodiment, the queuing/dequeuing module **240** of the NIC **160** receives the modified packets via a full-mesh interconnect (not shown) so that it can funnel packets originally received at the input ports of any NIC **160**, **180** installed within the communication device **150**, including the packets received by the input ports **162**, **164**, **166** of its own NIC **160**, to one or more of the output ports **168**, **170**, **172** of its own NIC **160**. In another embodiment, packets received at input ports **162**, **164**, **166** are transferred directly to the queuing/dequeuing module **240** by the forwarding engine **220**.

[0031] With reference to **FIGS. 3 and 4**, an illustrative embodiment of the structure of the forwarding engine **220** comprises ingress logic **310**, an ALE interface **350**, a statistics interface **360**, egress logic **370**, and one or more processors representatively shown at **320**, **330**, **340**. In operation, data corresponding to a packet is transmitted over communications network **110** and is received at a particular input port **162**, **164**, or **164** of NIC **160** or **180** that is coupled to the communications network **110** (step **410**). A processor **330** is then selected from a pool of processors (representatively indicated at **320**, **330**, **340**) associated with the input port **162**, **164**, or **166** to process the packet (step **420**). Once the processor **330** has been allocated, the packet is split into header and body portions by the ingress logic **310** (step **430**). The packet header is written into a particular location within a register file **710** (**FIG. 7**) associated with the processor **330**

using direct register accesses and the packet body is written to an output buffer in the egress logic **370** using direct memory accesses (step **440**). The processor **330** then processes the packet header according to locally stored instructions (step **450**) and transfers the processed packet header to the egress logic **370** where it is reintegrated with the packet body (step **460**).

[0032] The processor **330** may perform such tasks as processing the packet header by checking the integrity of the packet header, verifying its checksum, accessing the statistics module **230** via the statistics interface **360** to provide statistics that are used to report the processing activity involving this packet header to modules external to the forwarding engine **220**, and communicating with the ALE **210** via the ALE interface **350** to obtain routing information for one of the output ports **168, 170, 172** associated with the destination of the packet. Additional network specific (e.g., IP, ATM, Frame Relay, HDLC, TDM) packet processing may be done at this time. At the conclusion of this processing activity, the processor **330** modifies the packet header to include routing information (e.g., by prepending a forwarding vector to the packet header) that designates a particular output port **168, 170, 172** of the NIC **160**. The modified packet header is then written to the egress logic **370** of the forwarding engine **220** where it is subsequently routed to the queuing/dequeuing module **240** as discussed above.

[0033] The ALE Interface **350**, Statistics Interface **360** and egress logic **370** are resources within the forwarding engine **220** that are shareable among the processors **320, 330, 340**. An arbitration mechanism (not shown) is provided in the forwarding engine **220** to arbitrate between the processors **320, 330, 340** for access to these resources **350, 360, 370**. In one embodiment, when the processor **330** is allocated to the packet, a processor identifier, such as the processor number, for the processor **330** is communicated to each of the three shared resources **350, 360, 370** identified above. Each of these shared resources **350, 360, 370** then writes the processor number to a FIFO, which preferably has a depth equal to the total number of processors in the forwarding engine **220**. Logic in each of the shared resources **350, 360, 370** accesses its respective FIFO to determine which of the processors **320, 330**, or **340** should be granted access to the resource next. Once the granted processor completes its access to a particular resource **350, 360, 370**, the accessed resource reads its next FIFO entry to determine the next processor to which a grant will be issued.

[0034] More particularly and with reference to **FIGS. 5 and 6**, the receipt, manipulation, and transfer of the packet data within the forwarding engine **220** is handled primarily by a plurality of DMA and DRA state machines. In one illustrative embodiment, these state machines are contained within the ingress logic **310** and the processor **330**. During the operation of this illustrative embodiment, a packet is received from one of the input ports **162, 164, 166** of the NIC **160** and stored within a Receive_Data FIFO (First In/First Out buffer) **510** in the Ingress Logic **310** (step **610**). A Receive_Status FIFO **512** records the particular input port **162, 164**, or **164** at which the packet arrived and maintains an ordered list of input port numbers for each packet received by the forwarding engine **220**, which is sorted in accordance with when the packet was received.

[0035] An Issue_DMA_Command state machine **514** detects when the Receive_Status FIFO **512** contains data and acquires the input port number associated with the input port **162, 164**, or **166** that received the packet from the Receive_Status FIFO **512** (step **620**). The Issue_DMA_Command state machine **514** then sends a processor allocation request that contains the port number of the packet to an Allocate_Processor state machine **516**, which accesses an Allocation_Pool Register **518** associated with that port number to determine a set of processors **320, 330, 340** that are candidates to operate on this packet (step **630**). The Allocate_Processor state machine **516** then accesses a Processor_Free Register **520** to determine if any of the candidate processors **320, 330, 340** identified by the Allocation_Pool Register **518** are available for use. The Allocate_Processor state machine **516** subsequently allocates one of the available processors **330** from the set of candidate processors **320, 330, 340** to process the packet (step **640**) and sends the allocation grant and processor number of that processor **330** to the Issue_DMA_Command state machine **514**.

[0036] Upon receipt of the processor number associated with the allocated processor **330**, the Issue_DMA_Command state machine **514** sends an execute signal/command that contains the processor number to a DMA_Execute state machine **522**, which accesses a Header_DMA_Length Register **524** to obtain the amount of the received packet that is to be sent to the processor **330** (i.e., the length of the packet header) (step **650**). The DMA_Execute state machine **522** then issues a DMA command, which retrieves the header portion of the packet (corresponding to the packet header) from the Receive_Data FIFO **510** and transfers it on a DRA bus **526** where it is received by a Processor_DRA state machine **530** contained within the processor **330** (step **660**). The DMA_Execute state machine **522** also issues a DMA command that retrieves the packet body from the Receive_Data FIFO **510** and transfers it on another DMA bus **528** for receipt by a buffer (not shown) of the egress logic **370** (step **660**). The Processor_DRA state machine **530** subsequently writes the packet header data received via the DRA bus **526** directly to a register file region starting at a fixed address location (e.g., address **0**) in the register file space **710** (**FIG. 7**) of processor **330** (step **670**). The processor **330** then processes the packet header (Step **680**) and transmits the processed header to the egress logic **370** for reintegration with the packet body (step **690**) via the Transmit_DMA state machine **532**.

[0037] More particularly and with reference to **FIGS. 7 and 8**, the processing of the packet header in processor **330** is preferably such that the processor's instructions and activities are confined to the manipulation of data and execution results in the execution space formed within the processor's local register file **710**. The structure of the processor **330** in one illustrative embodiment comprises the Stats_Interface state machine **704**, the ALE_Interface State Machine **706**, the Processor_DRA state machine **530**, the Transmit_DMA state machine **532**, the register file **710**, an arithmetic logic unit (ALU) **720**, a processor control module **730**, and an instruction memory **740**. The computational unit **725** is comprised of the processor control **730** and the ALU **720**.

[0038] During the operation of this illustrative embodiment and while the processor **330** is awaiting receipt of a packet header, the computational unit **725** continually executes an instruction at a special address (e.g., address **0**) in the instruction memory **740** (i.e., in an infinite loop) (step

810). Hardware in the processor **330** detects address **0** to be a special address in which the instruction is returned from "hard-wired" instruction values etched in silicon rather than from instructions stored in instruction memory **740**. In one possible implementation, accessing the instruction at special address **0** returns a "JMP **0**" (or jump to address **0** instruction), thereby causing the processor **330** to execute an infinite loop at that address.

[0039] When a packet header is transferred to the processor's register file **710** from the ingress logic **310**, a control signal from the Processor_DRA state machine **530** indicates to the processor control module **730** that the packet header transfer is in progress (step **820**). While this signal is active, the processor control module **730** forces a processor program counter (not shown) to specify a non-special address (e.g., address **2**) of the instruction memory **740** and thus cause the computational unit **725** to break out of the infinite loop being executed at special address **0** and wait until the signal becomes inactive (step **830**). The computational unit **725** begins execution of the instruction at address **2** in response to the signal becoming inactive (step **840**). Address **2** of the instruction memory **740** can be configured to hold the first instruction that will be used to process the packet header in the register file **710** (i.e., the instruction at address **2** corresponds to the beginning of the "real" software image that has been previously downloaded to operate on packet headers). When the Processor_DRA state machine **530** completes the writing of the packet header beginning at a fixed location in the register file **710** (occurring when the control signal goes inactive), the computational unit **725** continues to normally execute the remaining instructions (i.e., beyond address **2**) in the instruction memory **740**. Specific instructions in the instruction memory **740** specify locations within the register file **710**. Upon completion of the processing activity on a particular packet header, the executing software "jumps" to address **0**, thus executing the instruction at address **0** in an infinite loop. This technique illustrates one particular implementation of how the processor **330** can be triggered to process the packet header stored in the register file **710** without using load and store instructions.

[0040] In another embodiment, the allocated processor **330** remains idle (i.e., not accessing instruction memory or executing instructions) until it receives a signal from an external state machine indicating that the register file **710** has been populated with the complete packet header. The computational unit **725** then executes code from instruction memory **740** to process the packet header. Triggering events can, for example, include when a control signal goes inactive. Alternatively, the allocated processor **330** is triggered when the DRA transfer has been initiated, completed, or when it is in process. Numerous other triggering events will be apparent to those skilled in the art.

[0041] As discussed earlier, the processor **330** accesses one or more shared resources (e.g. see **FIG. 3**, ALE Interface **350**, Statistics Interface **360**, and egress logic **370**) that are external to the processor **330** during the processing of the packet header. For example, the processor **330** interacts with the ALE **210** (**FIG. 2**) via the ALE Interface **350** (**FIG. 3**) to issue searches of the ALE **210** and to receive search results therefrom. These interactions with the ALE **210** performed by the processor **330** also occur without the processor **330** having to execute load and store instructions.

[0042] In one aspect and while executing the instructions in the instruction memory **740**, the processor **330** composes a search key starting at a predefined address in the register file **710**. The computational unit **725** executes an instruction which involves writing a value to the ALE_Command Register that specifies the amount of search key data to transmit to the ALE **210**. This value effectively serves as a control line to the ALE_Interface state machine **706** of the processor **330** and thus triggers the ALE_Interface state machine **706** to read the value or other data from the ALE_Command Register, to determine the amount of data to be transferred, and to transfer the specified data to the ALE Interface **350** using direct memory accesses that are independent of the computational unit **725**. While the processor **330** awaits the results of the search to be returned, it can perform other functions, such as verifying the network protocol (e.g., IP) checksum of the packet header. When the search results from the ALE **210** are available, they are transmitted to the ALE_Interface state machine **706** via the ALE Interface **350**. The ALE Interface state machine **706** writes the search results to a predetermined location of the register file **710** using one or more direct register accesses and signals the computational unit **725** when the write is complete. The computational unit **725** subsequently modifies the packet header in response to the search results.

[0043] The processor **330** can also issue a statistics update command by writing an address and length value into the Statistics Update_Command Register (not shown) of the processor **330**. The Statistics_Interface state machine **704** of the processor **330** is triggered to read the data from the Statistics_Update_Command Register, to determine the source and amount of data to transfer, and to transfer the specified data to the Statistics Interface **360** using direct memory accesses that are independent of the computational unit **725**.

[0044] Similarly, when the processor **330** has completed processing the packet header, the computational unit **725** writes the processed packet header to the Transmit_DMA state machine **532** of the processor **330**, which transfers the processed header to a buffer in the egress logic **370** using direct memory accesses that are independent of the processor **330** (step **850**). After all processing is complete, the software executing in processor **330** jumps back to address **0** of the instruction memory **740** and begins executing the infinite loop instructions discussed previously while waiting for the next packet header to arrive (step **860**).

[0045] More particularly, upon completion of the processing activity, the packet header may not necessarily reside in a contiguous region of the register file **710** and thus the computational unit **725** may have to specify the location of each piece of the processed packet header in the register file **710**. Accordingly, the computational unit **725** issues one or more writes to a Move DMA_Command Register (not shown) that specify the start address and length of each piece of the processed packet header. These writes are stored in a FIFO, essentially as a list of reassembly commands. After the data for all of the pieces of the fragmented packet header are obtained, the computational unit **725** writes to a Transmit_DMA_Command Register (not shown) and specifies the body length of the packet along with other data.

[0046] The value written to the Transmit_DMA_Comand Register triggers the Transmit_DMA state machine **532**

within the processor **330** to begin assembly of the packet header in accordance with the reassembly commands stored in the FIFO referenced above. The Transmit_DMA state machine **532** then transmits the assembled packet header, along with some control information (including the length of the packet body), to the egress logic **370** using direct memory accesses that are independent of the computational unit **725**. The egress logic **370** concatenates the processed packet header received from the Transmit_DMA state machine **532** with the packet body stored in a FIFO of the egress logic **370** and subsequently transmits the reconstituted packet to the queuing/dequeuing module **240** as previously discussed.

[0047] In order to properly reconstitute the packet header with the packet body, the processor **330** obtains the length of the overall packet from data embedded within the packet header itself and obtains the length of the packet header from data transferred to the processor **330** by the Receive_Data FIFO **510** (**FIG. 5**) (corresponding to the same value that was written to the Header_Length Register **524** of **FIG. 5**). Based upon this information, the processor **330** calculates the amount of packet body data that was previously transferred to the output FIFO in the egress logic **370** and specifies the length of the packet body as control information to be transmitted to the egress logic **370** by the Transmit_DMA state machine **532**. In this manner, the processor **330** is able to specify the amount of packet body data to pull from the output FIFO of the egress logic **370** that will be appended to the newly-assembled packet header formed by the processor **330** to reconstitute the modified packet. In order to properly reconstitute the modified packet, the processor **330** is granted access to the egress logic **370** in the same order in which the processor **330** was allocated (and thus in the same order as packet bodies were written to the output FIFO of the egress logic **370**).

[0048] Aspects of the present invention afford great flexibility in the assignment of compute resources to input packet processing requirements. Assuming for illustrative purposes that there are a total of **40** processors **320, 330, 340** within the forwarding engine **220**, the processors **320, 330, 340** can be flexibly allocated to meet the packet processing needs of a multitude of input/output port configurations. For example, in a NIC **160** where there is only a single logical input port (i.e., port **0**), all 40 processors **320, 330, 340** could be allocated to process packets for that single port. In this scenario, the code image loaded into the instruction memory **740** of each processor **320, 330, 340** could be identical, thus allowing each processor **320, 330, 340** to perform identical algorithms for that one type of input port. In another scenario involving four logical input ports, each with a different type of network interface, the processing algorithms required for each type of network interface could differ. In this case, the forty processors could be allocated as follows: processors **[0-9]** to port **0**, processors **[10-19]** to port **1**, processors **[20-29]** to port **2** and processors **[30-39]** to port **3**. In addition, four different code images could be downloaded, where each unique image corresponds to a particular input port. In yet another scenario, the NIC **160** may include two logical input ports, each with different processing performance requirements. In such a scenario, one of the input ports may consume 75% of the ingress bus bandwidth and have a packet arrival rate requiring 75% of the processor resources, with the second port accounting for the remainder. To support these performance requirements, thirty processors could be allocated to input port **0** and ten processors to input port **1**.

[0049] The programming model for NICs **160, 180** that incorporate multiple processors as part of their forwarding engines **220**, can be simplified by allocating a single processor to each packet received. Further, and as discussed above, the decreased die size realized by systems that incorporate the present invention allow the inclusion of additional processors in the forwarding engine ASICs of the NICS **160, 180**, which thereby ensure that packets can be transmitted at the wire rate of the network **110**. The present invention is readily scaleable by adding more processors on a given forwarding engine ASIC, increasing the clock rate of the processors, and by aggregating the processing pools of multiple ASICs. Note that in providing this capability, the hardware architecture of the invention maintains the packet order of the packets arriving via the network interface so that the reintegrated packets can be transmitted out of the forwarding engine in the appropriate order.

[0050] The processor pool aggregation technique may be particularly advantageous where the NIC **160** of the communication device **150** receives a packet data stream via the communication network **110** at a line rate that might otherwise overwhelm the processing capabilities of the NIC **160** and result in dropped packets and reduced quality of service. The aggregation technique allows the allocation of idle processors from more than one forwarding engine. For example, the NIC **160** may contain a plurality of forwarding engine ASICs, each with a pool of processor that can be allocated to process packets arriving at any input port on the NIC **160**. Alternatively, a pool of processors in additional forwarding engine ASICS, which are present on other NICs **180** within the communication device **150** can be allocated to the NIC **160** that is experiencing the heavy network load.

[0051] Although the present invention has been described with reference to specific details, it is not intended that such details should be regarded as limitations upon the scope of the invention, except as and to the extent that they are included in the accompanying claims.

What is claimed is:

1. A method of processing a packet, the method comprising the steps of:

receiving the packet;

identifying a packet header portion of the data packet;

transferring the packet header to a register file accessible to a processor; and

processing the packet header without invoking at least one of a load instruction and a store instruction by the processor.

2. The method of claim 1, wherein the transferring step is performed without invoking at least one of a load instruction and a store instruction.

3. The method of claim 1 further comprising the steps of:

splitting the packet into the packet header portion and a packet body portion;

transferring the packet header to the register file using a direct register access; and

transferring the packet body to an output buffer.

4. The method of claim 3 further comprising the steps of:

selecting an output port for transmission of the packet;

integrating the processed packet header with the packet body in the output buffer; and

forwarding the integrated packet from the output buffer to the selected output port for transmission therefrom.

5. The method of claim 1 further comprising the steps of:

providing a plurality of identical processors executing a common instruction set, each processor storing the instruction set locally to the processor;

selecting a processor from among the plurality to process the packet header; and

causing the selected processor to process the packet header.

6. The method of claim 5, wherein the step of selecting the processor is performed by a state machine responsive to the receipt of the packet at an input port.

7. The method of claim 5, wherein the step of causing the selected processor to process the packet header is performed by at least one state machine configured to write the packet header to at least one fixed location in the register file accessible to the selected processor.

8. The method of claim 5, further comprising the step of downloading a common instruction set to an instruction memory in each of the plurality of processors.

9. A method of processing a packet header of a packet received over a communications network, the method comprising the steps of:

transferring the packet header to at least one fixed location in a register file;

providing a processor associated with the register file, the processor repetitively executing an instruction in an infinite loop, the instruction being stored at a first known location in an instruction memory associated with the processor;

causing the processor to execute instructions beginning at a second known location in the instruction memory responsive to the transfer of the packet header;

processing the packet header in the at least one fixed location in the register file in accordance with the instructions beginning at the second known location in the instruction memory; and

resetting the processor to repetitively execute the instruction stored at the first known location in the instruction memory upon completing the processing of the packet header.

10. The method of claim 9, wherein the processing step comprises processing the packet header without invoking at least one of a load instruction and a store instruction.

11. The method of claim 9, further comprising the steps of

receiving the packet at an input port coupled to the communications network;

selecting the processor from a plurality of candidate processors associated with the input port;

splitting the packet into the packet header and a packet body; and

transferring the packet header to the at least one fixed location in the register file associated with the selected processor by executing a DRA command issued by a state machine coupled to the register file.

12. The method of claim 11, further comprising the step of downloading a common instruction set to an instruction memory in each of the plurality of candidate processors.

13. A packet-processing system for processing a packet received over a communications network, the system comprising:

an input port configured to receive the packet over the communications network;

a processor associated with the input port;

a register file accessible to the processor; and

an ingress element coupled to the input port, processor, and register file, the ingress element being configured to transfer at least one portion of the packet to the register file by invoking a DRA command,

wherein the processor processes the at least one portion of the packet in the register file in response to the DRA command and without invoking at least one of a load instruction and a store instruction.

14. The packet-processing system of claim 13, wherein the ingress element is configured to select the processor from a plurality of candidate processors associated with the input port.

15. The packet-processing system of claim 14, further comprising a plurality of instruction memories, each of the plurality of instruction memories being associated with a corresponding one of the plurality of candidate processors, wherein the plurality of instruction memories contain an identical instruction set.

16. The packet-processing system of claim 13, wherein the at least one portion of the packet corresponds to a header of the packet.

17. The packet-processing system of claim 16, wherein the ingress element comprises a state machine configured to write the packet header to a fixed location in the register file.

18. A packet-processing system for processing a packet header of a packet received over a communications network, the system comprising:

an input port coupled to the communications network;

an ingress element coupled to the input port and configured to receive and parse the packet to obtain the packet header;

a register file coupled to the ingress element and configured to store the packet header received from the ingress element at an at least one fixed location;

an instruction memory configured to return instructions from at least a first and second address; and

a processor coupled to the ingress element, register file, and instruction memory, the processor repetitively executing instructions stored at the first of the instruction memory, wherein the processor executes instructions beginning at the second address of the instruction memory to process the packet header in the register file in response to a signal from the ingress element.

19. An information-processing system comprising:

a processor having an internal register file space and a unit for manipulating data;

an ingress element for delivering unprocessed data to the internal register file space; and

an egress element for removing processed data from the internal register file space,

wherein operation of the processor is confined to manipulating data within the internal register file space.

20. The system of claim 19 further comprising at least one state machine governing operation of the ingress and egress elements and responsive to instructions within the internal register file space, the state machine causing data to be moved into and out of the internal register file space using direct accesses thereto in accordance with the instructions.

21. The system of claim 20 further comprising a network interface receiving data from a communications network, the interface providing received data to the ingress element.

22. A method of information processing, the method comprising the steps of:

providing a processor having an internal register file space and a unit for manipulating data; and

delivering unprocessed data to the internal register file space and removing processed data from the internal register file space using direct accesses to the internal register file space, operation of the processor being confined to manipulating data within the internal register file space.

23. The method of claim 22 further comprising the steps of:

providing at least one state machine governing delivery of data to and removal of data from the internal register file space using direct accesses thereto; and

causing the processor to signal the state machine by writing a value into a control register, the state machine being responsive to the value and carrying out the direct accesses in accordance with a state machine logic.

24. The method of claim 22 wherein the unprocessed data originates with a communications network having a line data rate, the processor processing data at a rate equal to the line rate.

25. The method of claim 24 wherein the unprocessed data is in the form of packets.

26. A method of processing a packet stream comprising a temporal sequence of packets, the method comprising the steps of:

providing a plurality of identical processors executing a common instruction set, each processor storing the instruction set locally to the processor;

receiving the packets;

for each packet, (i) identifying a packet header portion of the data packet, (ii) selecting a processor from among the plurality to process the packet header based on processor availability, and (iii) causing the selected processor to process the packet header using the locally stored instructions; and

assembling the processed packets to reconstruct the packet stream in accordance with the temporal sequence.

27. The method of claim 26 wherein the plurality of processors is physically located on a plurality of integrated circuits.

28. A system for processing a packet stream comprising a temporal sequence of packets, the system comprising:

a plurality of identical processors executing a common instruction set, each processor comprising a local instruction memory containing the instruction set;

an input port for receiving the packets;

an ingress logic unit coupled to the input port and to the processors, the ingress logic unit being configured, for each packet, to (i) identify a packet header portion of the data packet and (ii) select a processor from among the plurality to process the packet header based on processor availability, the selected processor responding to the ingress logic unit by processing the packet header using the locally stored instructions; and

an egress logic unit for assembling the processed packets to reconstruct the packet stream in accordance with the temporal sequence.

29. The system of claim 28 wherein the plurality of processors is physically located on a plurality of integrated circuits.

* * * * *

RELATED PROCEEDINGS APPENDIX

As stated above, as presently advised, there are no other prior or pending appeals, interferences, or judicial proceedings known to Appellant, the Appellant's legal representative, or Assignee which may be related to, directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal. Thus, no copies of decisions rendered by a court or by the Board are provided.

Respectfully submitted,

07/20/2011
Date

Ross D. Snyder, Reg. No. 37,730
Attorney for Appellant(s)
Ross D. Snyder & Associates, Inc.
PO Box 164075
Austin, Texas 78716-4075
(512) 347-9223 (phone)
(512) 347-9224 (fax)

US005802178A

United States Patent [19]

Holden et al.

[11] Patent Number: 5,802,178

[45] Date of Patent: Sep. 1, 1998

[54] **STAND ALONE DEVICE FOR PROVIDING SECURITY WITHIN COMPUTER NETWORKS**

[75] Inventors: **James M. Holden**, Valley Center; **Stephen E. Levin**, Poway, both of Calif.; **James O. Nickel**, Dayton, Md.; **Edwin H. Wrench, Jr.**, San Diego, Calif.

[73] Assignee: **ITT Industries, Inc.**, White Plains, N.Y.

[21] Appl. No.: **688,525**

[22] Filed: **Jul. 30, 1996**

[51] **Int. Cl.**[6] **H04L 9/00**

[52] **U.S. Cl.** **380/49**

[58] **Field of Search** 380/49, 9, 23

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,228,083	7/1993	Lozowick et al.	380/9
5,602,920	2/1997	Bestler et al.	380/49

Primary Examiner—David C. Cain
Attorney, Agent, or Firm—Plevy & Associates

[57] **ABSTRACT**

A multi-level security device is disclosed for providing security between a user and at least one computer network, wherein the user is selected from the group consisting of a host computer and at least a second network. A secure network interface Unit (SNIU) that operates at a user layer communications protocol, which communicates with other like SNIU devices by establishing an association at a session layer of a communication stack in order to create a global security perimeter for end-to-end communications. The SNIU includes a host/network interface for receiving messages sent between the user and the at least one network, which is operative to convert the received messages to and from a format utilized by the at least one network. A message parser for determining whether the association already exists with another SNIU device. A session manager coupled to the interface for identifying and verifying the user requesting access to the network. The session manager also for transmitting the messages received from the user when the message parser determines the association already exists. An association manager coupled to the interface for establishing an association with other like SNIU devices when the message parser determines the association does not exist.

19 Claims, 6 Drawing Sheets

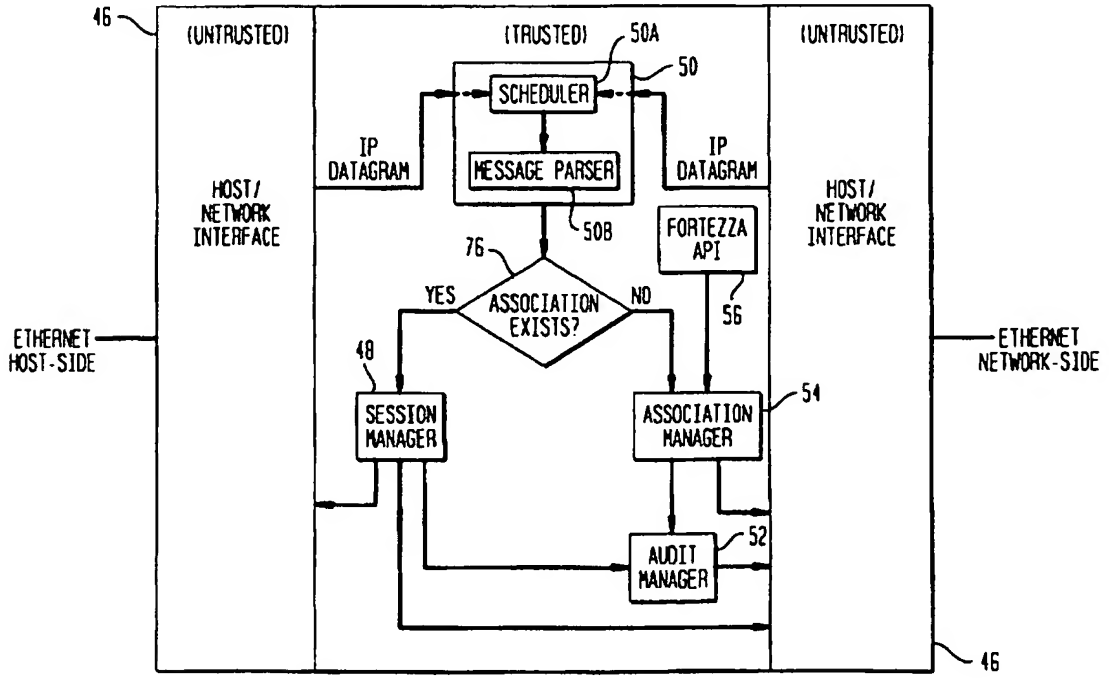
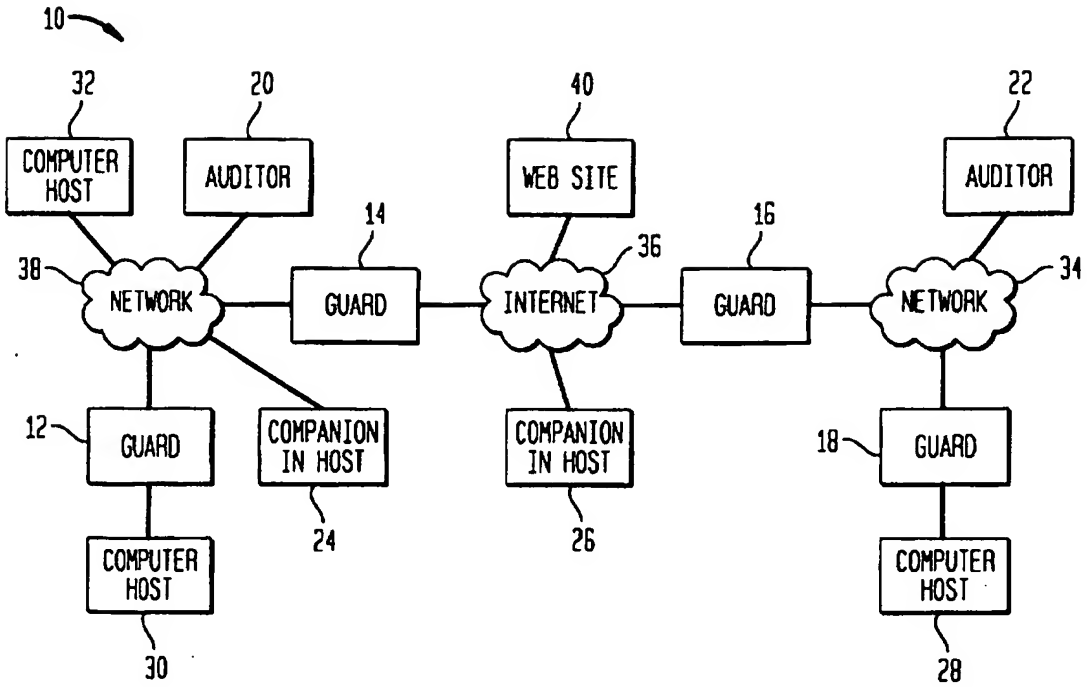


FIG. 1


10
32
COMPUTER HOST
20
AUDITOR
40
WEB SITE
22
AUDITOR
14
16
38
NETWORK
GUARD
36
INTERNET
GUARD
NETWORK
34
12
GUARD
COMPANION IN HOST
COMPANION IN HOST
18
GUARD
24
26
COMPUTER HOST
COMPUTER HOST
30
28

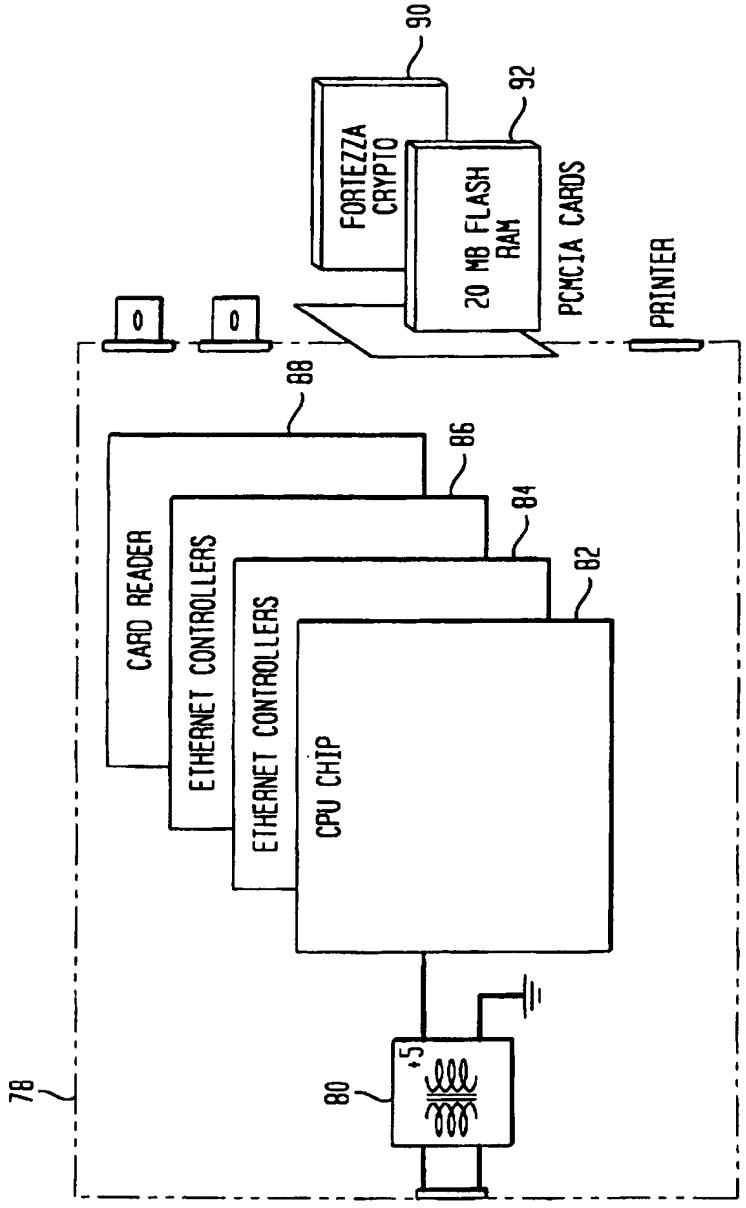

90
92
FORTEZZA CRYPTO
20 MB FLASH RAM
PCMCIA CARDS
PRINTER
88
86
84
82
CARD READER
ETHERNET CONTROLLERS
ETHERNET CONTROLLERS
CPU CHIP
+5
80
78

FIG. 2

FIG. 3

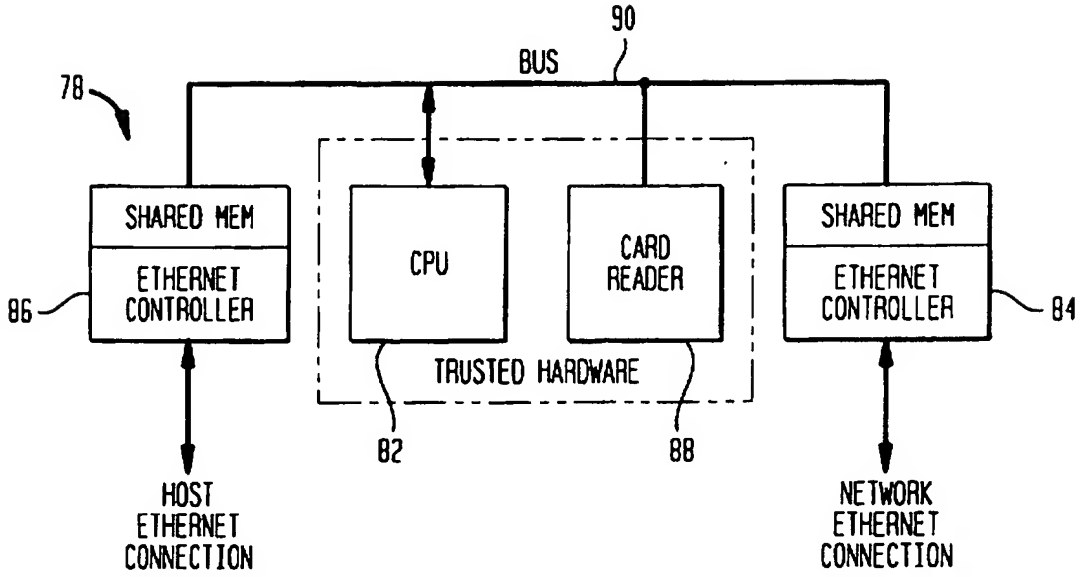

90
BUS
78
SHARED MEM
ETHERNET CONTROLLER
86
CPU
CARD READER
TRUSTED HARDWARE
82
88
SHARED MEM
ETHERNET CONTROLLER
84
HOST ETHERNET CONNECTION
NETWORK ETHERNET CONNECTION

FIG. 4

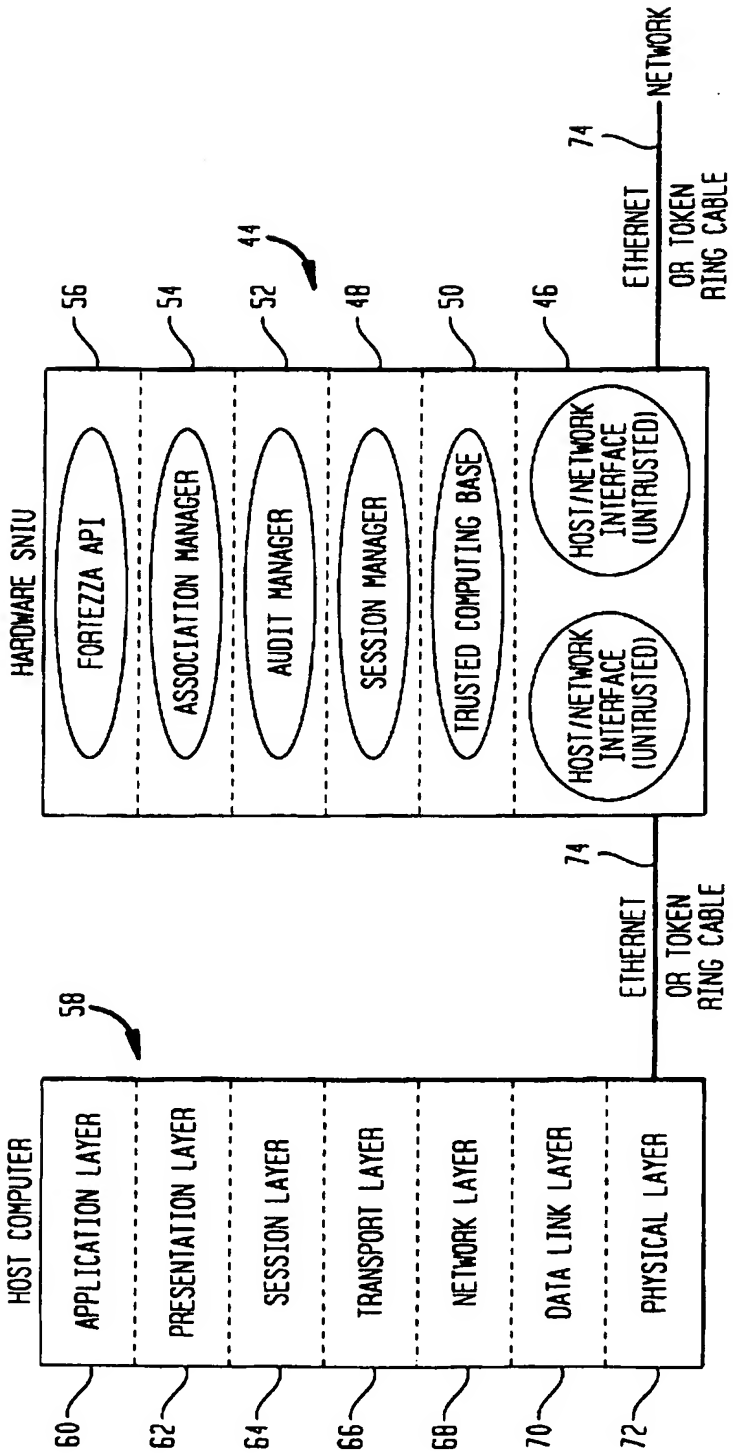

HOST COMPUTER
APPLICATION LAYER
PRESENTATION LAYER
SESSION LAYER
TRANSPORT LAYER
NETWORK LAYER
DATA LINK LAYER
PHYSICAL LAYER
58
60
62
64
66
68
70
72
ETHERNET OR TOKEN RING CABLE
74
HARDWARE SNIU
FORTEZZA API
ASSOCIATION MANAGER
AUDIT MANAGER
SESSION MANAGER
TRUSTED COMPUTING BASE
HOST/NETWORK INTERFACE (UNTRUSTED)
HOST/NETWORK INTERFACE (UNTRUSTED)
44
46
48
50
52
54
56
ETHERNET OR TOKEN RING CABLE
74
NETWORK

FIG. 5

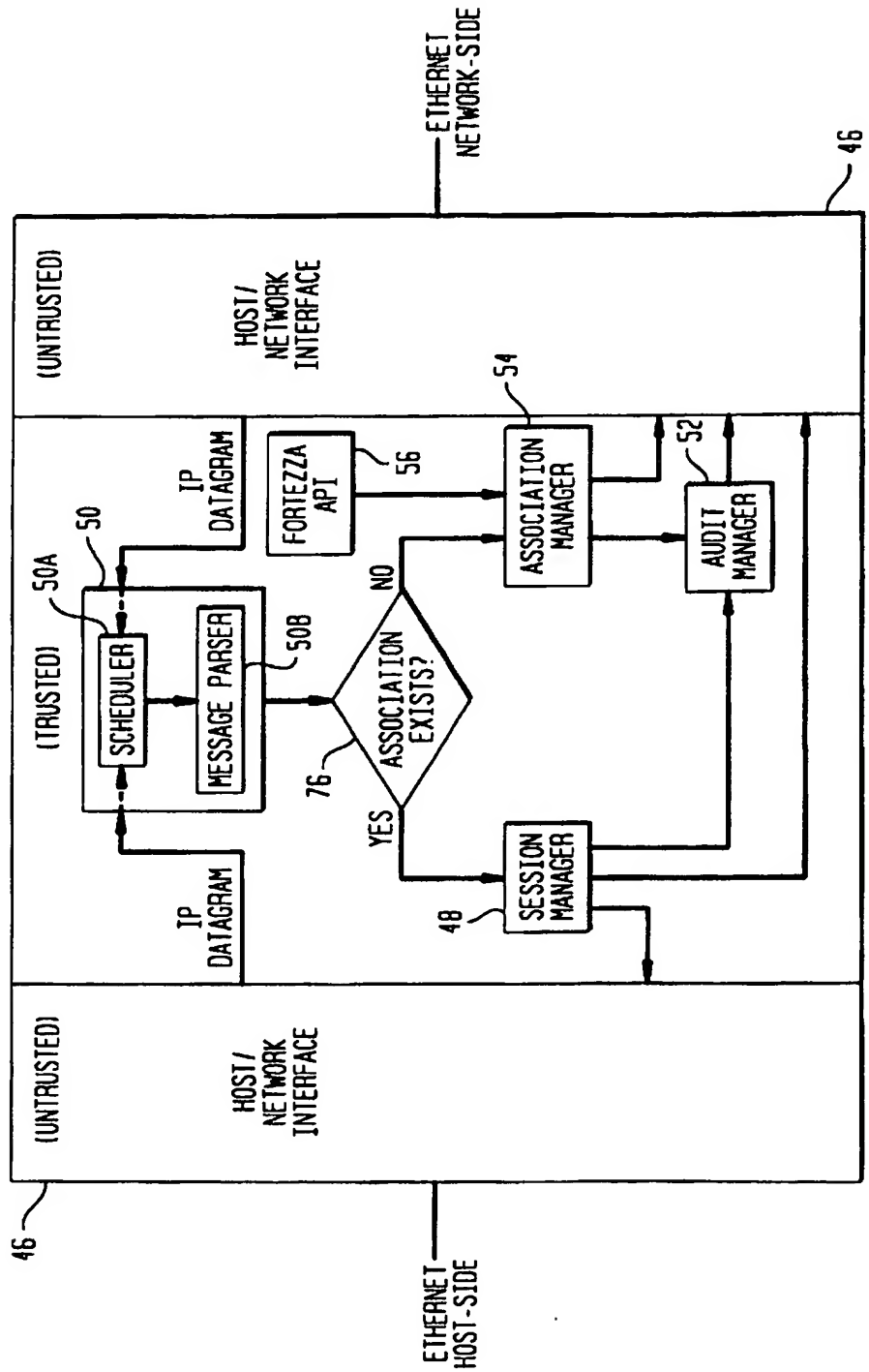

ETHERNET NETWORK-SIDE
46
(UNTRUSTED)
HOST/ NETWORK INTERFACE
54
52
IP DATAGRAM
FORTEZZA API
56
ASSOCIATION MANAGER
AUDIT MANAGER
50
50A
(TRUSTED)
SCHEDULER
MESSAGE PARSER
50B
NO
ASSOCIATION EXISTS?
76
YES
IP DATAGRAM
48
SESSION MANAGER
(UNTRUSTED)
HOST/ NETWORK INTERFACE
46
ETHERNET HOST-SIDE

FIG. 6

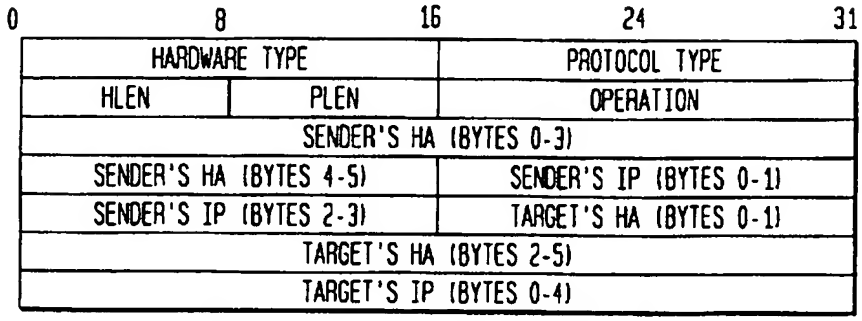


FIG. 7

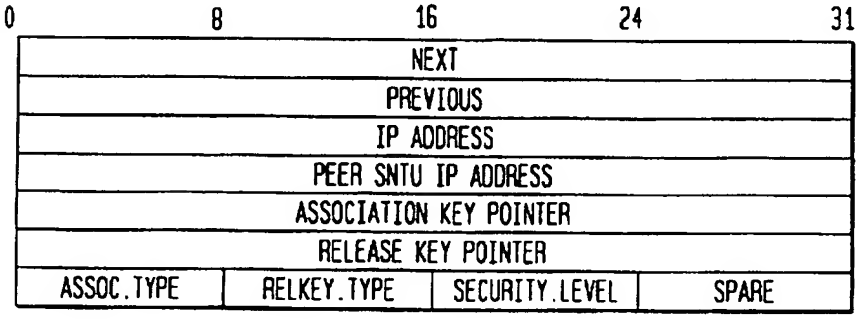
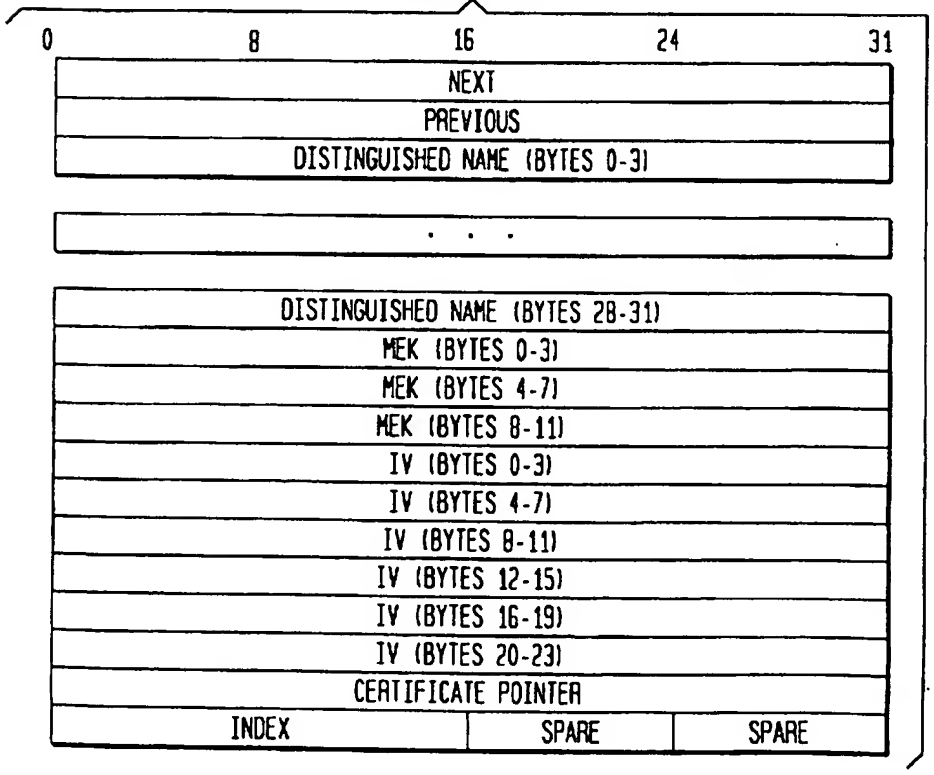


FIG. 8



STAND ALONE DEVICE FOR PROVIDING SECURITY WITHIN COMPUTER NETWORKS

RELATED APPLICATIONS

The Assignee herein, ITT Corporation, is the record owner of co-pending U.S. application Ser. No. 08/270,398 to Boyle et al., entitled APPARATUS AND METHOD FOR PROVIDING NETWORK SECURITY, filed Jul. 5, 1994.

FIELD OF THE INVENTION

The present invention relates in general to secure and multi-level secure (MLS) networks and in particular to a stand alone device for providing security and multi-level security for computer hosts utilized in secure and non-secure networks.

BACKGROUND OF THE INVENTION

Multi-level secure (MLS) networks provide a means of transmitting data of different classification levels (i.e. Unclassified, Confidential, Secret and Top Secret) over the same physical network. To be secure, the network must provide the following security functions: data integrity protection, separation of data types, access control, authentication and user identification and accountability.

Data integrity protection ensures that data sent to a terminal is not modified enroute. Header information and security level are also protected against uninvited modification. Data integrity protection can be performed by check sum routines or through transformation of data, which includes private key encryption and public key encryption.

Separation of data types controls the ability of a user to send or receive certain types of data. Data types can include voice, video, E-Mail, etc. For instance, a host might not be able to handle video data, and, therefore, the separation function would prevent the host from receiving video data. The system should include sequential review prior to data release where a plurality of users would review the data to approve release prior to actual release and the use of data type to separate management type data from ordinary user traffic.

Access control restricts communication to and from a host. In rule based access control, access is determined by the system assigned security attributes. For instance, only a user having Secret or Top Secret security clearance might be allowed access to classified information. In identity based access control, access is determined by user-defined attributes. For instance, access may be denied if the user is not identified as an authorized participant on a particular project. For control of network assets, a user may be denied access to certain elements of the network. For instance, a user might be denied access to a modem, or to a data link, or to communication on a path from one address to another address.

Identification of a user can be accomplished by a unique name, password, retina scan, smart card or even a key for the host. Accountability ensures that the a specific user is accountable for particular actions. Once a user establishes a network connection, it may be desirable that the user's activities be audited such that a "trail" is created. If the user's actions do not conform to a set of norms, the connection may be terminated.

Currently, there are three general approaches to providing security for a network: trusted networks, trusted hosts with trusted protocols, and encryption devices. The trusted network provides security by placing security measures within the configuration of the network. In general, the trusted network requires that existing protocols and, in some cases, physical elements be replaced with secure systems. In the Boeing MLS Lan, for instance, the backbone cabling is replaced by optical fiber and all access to the backbone is mediated by security devices. In the Verdix VSLAN, similar security devices are used to interface to the network, and the network uses encryption instead of fiber optics to protect the security of information transmitted between devices. VSLAN is limited to users on a local area network (LAN) as is the Boeing MLS Lan.

Trusted hosts are host computers that provide security for a network by reviewing and controlling the transmission of all data on the network. For example, the U.S. National Security Agency (NSA) has initiated a program called Secure Data Network System (SDNS) which seeks to implement a secure protocol for trusted hosts. In order to implement this approach, the installed base of existing host computers must be upgraded to run the secure protocol. Such systems operate at the Network or Transport Layers (Layers 3 or 4) of the Open Systems Interconnection (OSI) model.

Encryption devices are used in a network environment to protect the confidentiality of information. They may also be used for separation of data types or classification levels. Packet encryptors or end-to-end encryption (EEE) devices, for instance, utilize different keys and labels in protocol headers to assure the protection of data. However, these protocols lack user accountability since they do not identify which user of the host is using the network, nor are they capable of preventing certain users from accessing the network. EEE devices typically operate at the Network Layer (Layer 3) of the OSI model. There is a government effort to develop cryptographic protocols which operate at other protocol layers.

A number of network security products have been developed which include Raptor Eagle, Raptor Remote, Entrust, Secret Agent and Veil. Although, these products serve the same purpose, a number of different approaches have been utilized. For example, Raptor Eagle, Raptor Remote, and Veil implement these products as software instantiations. While Entrust and Secret Agent utilize hardware cryptographic components. Additionally, Raptor products are also application independent.

A problem with the above described products is that none are based upon the use of highly trusted software. Veil is an off-line encryption utility, which cannot prevent the inadvertent release of non-encrypted information. While Raptor Eagle and Raptor Remote are based on software instantiations and thus cannot be verified at the same level of assurance. Secret Agent and Entrust while hardware based are dependant upon the development of integration software for specific applications.

It is therefore, an object of the present invention, to provide a stand alone multi-level security device which provides an adequate level of security assurances for computer hosts utilized in secure as well as non-secure networks.

SUMMARY OF THE INVENTION

In accordance with the present invention, a network security apparatus and method for a network comprises a secure network interface unit (SNIU) coupled between host computer or user computer unit, which may be non-secure, and a network (i.e. a SNIU can be placed between two networks), which may be non-secure. When an SNIU is

implemented at each computer unit to be secured on the network, a global security perimeter is provided for ensuring security policy enforcement, controlled communication release, controlled communication flow, and secure session protocols through each computer unit interface.

In a preferred embodiment, a hardware SNIU is configured to process a defined trusted session protocol (TSP) and perform the core functions of host/network interface by utilizing an association manager, session manager and data sealer. The user/service interface function performs a standard communications stack function by handling all of the standard communications data translation between the Physical Data Link and Network protocol layers (i.e. layers one thru three). The host/network interface does not require the same level of trust as the rest of SNIU's software. This allows this software to be logically and physically separated from the rest of the software without effecting the underlying security of the system as a whole. The association manager functions include host computer and peer SNIU identification, audit, association setup and termination and maintenance of the sealer keys generated for the association between the two peer SNIUs. The session manager functions include sealing, verifying message authentication codes, audit and enforcing a security on each datagram passed thru the SNIU.

The above described SNIU is capable of communicating with other like SNIU devices creating a global security perimeter for end-to-end communications and wherein the network may be individually secure or non-secure without compromising security of communications within the global security perimeter.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic diagram of an MLS network system in accordance with the present invention;

FIG. 2 is a diagram of the hardware included in the hardware SNIU in accordance with the present invention;

FIG. 3 is a block diagram of the hardware SNIU in accordance with the present invention;

FIG. 4 is a block diagram of the hardware SNIU interfacing with the computer host in accordance with the present invention;

FIG. 5 is a data flow diagram for the hardware SNIU in accordance with the present invention;

FIG. 6 is a table showing the format for the Address Resolution Protocol and Reverse Address Resolution Protocol messages in accordance with the present invention;

FIG. 7 is a table showing the data structure for a token of the Association Table in accordance with the present invention; and

FIG. 8 is a table showing the data structure of a token for the Sym_Key Table in accordance with the present invention.

DETAILED DESCRIPTION

The present invention is directed to a secure network interface unit (SNIU), which is utilized to control communications between a user such as a computer host and a network at a "session layer" of interconnection which occurs when a user on the network is identified and a communication session is to commence. For example, the industry-standard Open Systems Interconnection (OSI) model, defines seven layers of a network connection: (1) physical; (2) data link; (3) network; (4) transport; (5) session; (6) presentation; and (7) application. In the present invention, the network security measures are implemented at the Session Layer 5. The placement of security at the Session Layer allows existing network assets and existing network protocols at the Transport Layer 4 and lower to continue to be used, thereby avoiding the need to replace an installed network base for the implementation of the multi-level security system. The connected host or user equipment and the network backbone are therefore not required to be secure (trusted). Conventionally, OSI network applications employ CCITT X.215 which is a non-secure session layer protocol. None of the prior multi-level security systems employ the security measures described herein in the Session Layer.

The SNIU according to the present invention can be configured in a number of different embodiments depending on the particular physical locations and forms of implementation.

These embodiments include a stand alone hardware SNIU and a software SNIU.

The hardware embodiment of the SNIU is implemented solely as a stand alone hardware device. Such a configuration is desirable, since the stand alone SNIU is highly trusted. The stand alone SNIU is configured to be inserted between existing hosts and a network. The SNIU is transparent to the host, and any legacy system or application software running on the host. The stand alone SNIU provides protection for any host connected to an IP based network. There is no requirement that the attached host computers run a trusted operating system. The stand alone SNIU provides a trusted boundary between the protected hosts and the unprotected network. Protected means that the connection is with another known SNIU (a unique digital signature identifies the SNIU), the messages are confidential (encrypted) and unaltered (cryptographic residues validate the packet).

The software embodiment of the SNIU is implemented solely as a software function resident in and executed from the host machine. Such a configuration is desirable, since the software SNIU is designed to be installed in existing portable computers, which avoids the additional cost of additional hardware required by a stand alone hardware SNIU. The software SNIU provides the same network security features as the stand alone SNIU when the host computer is connected to home enterprise's network. The software SNIU also extends that same level of security across the Internet (or any other unprotected network) when the user is on the road and is remotely communicating with the enterprise network or other remotely located computer devices including a similar software SNIU.

The software SNIU provides all of the functionality and security of the stand alone SNIU as well as complete operability with these devices. The software comprising the software SNIU is based on the same software utilized in the stand alone hardware SNIU. The user of the software SNIU assumes an acceptable risk in exchange for not requiring additional hardware required by a stand alone SNIU, which cannot be circumvented or corrupted via attacks from originating from external hardware. By providing reasonable software protection (not allowing unauthorized personal physical access) and software protection (anti-virus protection), a software SNIU can be utilized providing the user with an acceptable level of risk. If the user is confident that the software comprising the software SNIU is not circumvented or modified, then he can enjoy the same degree of confidence as the user of a stand alone device.

Referring to FIG. 1, there is shown an example of a Multi-Level Security (MLS) System in accordance with the

present invention. This system **10** incorporates the various embodiments of the SNIUs in order to provide MLS for computer networks such as the Internet. For example, the guard devices **14,16** which are hardware embodiments of the SNIU are coupled between computer networks **34,36,38** providing inter-network security. Additional guard devices **12,18** are coupled between users such as computer hosts **28,30,32** and the respective networks **30,32,34**. The software embodiment of the SNIU are implemented as companions within computer hosts **24,26**, which provides network security without requiring additional hardware. The auditors **20,22** are also hardware SNIUs which are configured to communicate directly with the other SNIUs to log audit events and potentially signal alarms. The above described system **10** enables secured and non-secured users such as a web site **40** to communicate with each other without the danger of compromising security.

During operation, the SNIUs included in the above described system **10** communicate with each other thereby creating a global security perimeter for end-to-end communications and wherein the network may be individually secure or non-secure without compromising security of communications within the global security perimeter. The SNIUs are capable of passing digital data, voice and video traffic so as to provide the full functionality required for a Trusted Session Protocol (TSP). The TSP uses the facilities of the lower level protocols to transmit data across the networks. To this end, and to provide flexibility, the specialized network interface SNIU is designed to allow coupling of the TSP with existing (non-secure) equipment and underlying network.

Security System Policies

The security system of the present invention may implement a number of security policies suitable to the circumstances of a given network environment. The major policy areas are: discretionary access control; mandatory access control; object reuse; labeling; identification and authentication; audit; denial of service detection; data type integrity; cascading control; and covert channel use detection.

Discretionary access control is a means of restricting access to objects (data files) based on the identity (and need to know) of the user, process, and/or group to which the user belongs. It may be used to control access to user interface ports based on the identity of the user. For a single-user computer unit, this mechanism may be implemented in the SNIU, whereas for a multi-user host, the DAC control may be implemented at the host machine. Discretionary access control may also be implemented as discretionary dialog addressing, wherein the addressing of all communications originated by a user is defined, and for user discretionary access denial, wherein a user may refuse to accept a communication from another user.

Mandatory access control is a means of restricting access to objects based on the sensitivity (as represented by a classification label) of the information contained in the objects, and the formal authorization (i.e., clearance) of the user to access information of such sensitivity. For example, it may be implemented as dialog lattice-based access control, wherein access requires a correct classification level, integrity level, and compartment authorization, dialog data-type access control, wherein correct data type authorization is required for access, and cascade protection, wherein controls are provided to prevent unauthorized access by cascading user access levels in the network.

Object reuse is the reassignment and reuse of a storage medium (e.g., page frame, disk sector, magnetic tape) that once contained one or more objects to be secured from unauthorized access. To be secured, reused, and assigned to a new subject, storage media must contain no residual data from the object previously contained in the media. Object reuse protection may be implemented by port reuse protection, session reuse protection, dialog reuse protection, and/or association reuse protection.

Labeling requires that each object within the network be labeled as to its current level of operation, classification, or accreditation range. Labeling may be provided in the following ways: user session security labeling, wherein each user session is labeled as to the classification of the information being passed over it; dialog labeling, wherein each dialog is labeled as to the classification and type of the information being passed over it; and host accreditation range, wherein each host with access to the secured network is given an accreditation range, and information passing to or from the host must be labeled within the accreditation range.

Identification is a process that enables recognition of an entity by the system, generally by the use of unique user names. Authentication is a process of verifying the identity of a user, device, or other entity in the network. These processes may be implemented in the following ways: user identification; user authentication; dialog source authentication, wherein the source of all communication paths is authenticated at the receiving SNIU before communication is allowed; SNIU source authentication, wherein the source SNIU is authenticated before data is accepted for delivery; and administrator authentication, wherein an administrator is authenticated before being allowed access to the Security Manager functions.

An audit trail provides a chronological record of system activities that is sufficient to enable the review of an operation, a procedure, or an event. An audit trail may be implemented via a user session audit, a dialog audit, an association audit, an administrator audit, and/or a variance detection, wherein audit trails are analyzed for variance from normal procedures.

Denial of service is defined as any action or series of actions that prevent any part of a system from functioning in accordance with its intended purpose. This includes any action that causes unauthorized destruction, modification, or delay of service. The detection of a denial of service may be implemented for the following condition: user session automatic termination, such as when unauthorized access has been attempted; user machine denial of service detection, such as detection of a lack of activity on a user machine; dialog denial of service detection; association denial of service detection, such as detection of a lack of activity between SNIUs; and/or data corruption detection, such as when an incorrect acceptance level is exceeded.

Covert channel use is a communications channel that allows two cooperating processes to transfer information in a manner that violates the system's security policies. Detection of covert channel use may be implemented, for example, by delay of service detection, such as monitoring for unusual delays in message reception, or dialog sequence error detection, such as monitoring for message block sequence errors.

Details of the Hardware SNIU

Referring to FIG. **2**, a diagram of the hardware included in the hardware SNIU in accordance with the present invention is shown. The SNIU **78** includes a Central Processing Unit (CPU) **82**, which may be a 486 CPU. The CPU **82** is utilized to run the trusted software of the SNIU **78**, which will be described in detail later. The SNIU **78** also includes two Ethernet controllers **84,86** which are preferably

PC/104 cards. The two Ethernet controllers 84,86 enable the SNIU 78 to be utilized over a standard Ethernet Local Area Network.

Also included is a card reader 88 which is preferably a standard PCMCIA device. The card reader 88 is adapted to receive a Fortezza card 90 and a Flash Ram card 92 which preferably has a capacity of twenty megabytes. The Fortezza card 90 is utilized to perform integrity and authenticating functions. While the Ram card 92 is utilized to load the SNIU software and associated security parameters. Power is supplied to the above described hardware by a five volt DC-DC converter 80.

Referring to FIG. 3, there is shown a block diagram describing the interconnection of the hardware SNIU according to the present invention. A commercial bus 90 which is preferably a PC-104 type is utilized to interconnect the CPU 82, Ethernet controllers 84,86 and card reader 88 as shown.

During operation, the trusted software in the CPU 82 preferably runs in the protected mode by making use of the memory and I/O protection provided by the 486 CPU. The software is partitioned into three independent tasks including the host side of the communications stack, network side of the communications stack and trusted security software. The two communications stacks are untrusted and limited by the 486 memory protection mechanisms to access each stacks memory partitions and I/O space of one of the Ethernet controllers 84,86. The untrusted code is able to access the untrusted code itself, a stack and scratch pad area, and a memory partition shared with trusted stacks. The untrusted code cannot access or change any memory other than these partitions. Thus, the shared memory partitions provide communications between the trusted and untrusted tasks.

By using the memory management of the CPU 82 to provide isolated memory enclaves, the amount of trusted code in the SNIU 78 is reduced. In addition, this enables commercial untrusted code to be utilized for both the communications stack and Ethernet controllers 84,86. The SNIU 78 can be configured to utilize either Ethernet controller 84,86 without changing any of the trusted code since all the network code is untrusted.

As previously described, the SNIU includes a card reader 88 adapted to receive a Fortezza card, which is utilized to perform integrity and authenticating functions. The integrity function is performed by encrypting data or messages leaving the SNIU 78 and decrypting data or messages received by the SNIU 78. The authentication function is performed by utilizing digital signatures. The Fortezza card includes a private key that writes a unique digital signature on all the outgoing messages of the SNIU 78. The Fortezza card also includes a public key for reading the digital signatures included in messages from other SNIU devices. By utilizing digital signatures unique to each SNIU device, the SNIUs are able to identify which other SNIU sent the message. This enables the SNIUs to authenticate that the message is coming from another SNIU which is a secure source and further enables it to determine which particular one sent the message.

Referring to FIG. 4, there is shown a block diagram of the hardware SNIU interfacing with a host computer. The SNIU includes trusted software 44 that interfaces with the communications stack of the host computer 58 through the Ethernet or token cable 74. The main modules of the SNIU include a Host/Network Interface 46, Session Manager 48, Trusted Computing Base 50, Audit Manager 52, Association Manager 54 and Fortezza API 56. The primary data structures included in the SNIU are the Association Table, Sym_Key Table, Certificate Table, Waiting Queue and Schedule Table. These data structures are described later in the description of the protocol. The communications stack of the computer host 58 is a typical OSI model including a physical 72, data link layer 70, network layer 68, transport layer 66, session layer 64, presentation layer 62 and application layer 60.

Referring to FIG. 5, a data flow diagram for the hardware SNIU is shown. When the host computer communicates with another computer over a network, the communications protocol stack within the computer processes the data to be transmitted. If a user on the computer is transmitting a file to another computer, the user may select the file to send by interacting with application layers software. The display which the user sees is controlled by presentation layer software. Session layer software checks the users permission codes to determine if the user has access to the file. Transport layer software prepares Internet Protocol Datagrams containing blocks of file data and determines that the transmitted file data is properly received and acknowledged or is re-transmitted.

The Host/Network Interface module 46 is utilized to intercept the packets of data and convert then back to IP datagrams (i.e., moves the data back up from the physical layer, through the data link layer, to the network layer). The interface 46 also processes Address Resolution Protocol (ARP) and Reverse Address Resolution Protocol (RARP) message and must coordinate the SNIU's two host/network interface ports hardware addresses with IP addresses of host computers on the other side of the SNIU.

When the untrusted Host/Network Interface 46 completes re-assembling an IP datagram from a host computer, the datagram is passed to the Trusted Computing Base 50 (TCB) of the SNIU for processing. The TCB 50 is the collection of hardware and software which can be trusted to enforce the security policy. The TCB 50 includes a Scheduler module 50A and a Message Parser 50B. The Scheduler 50A controls the hardware which controls access to memory and guarantees that IP datagrams are not passed directly from the host-side Host/Network Interface module to the network-side Host/Network Interface module 46 or vice versa. Rather each IP datagram is passed to the SNIU's other trusted software modules (Message Parser 50B, Association Manager 54, Session Manager 48) which determine if the IP datagram is allowed to pass through the SNIU and if it is encrypted/decrypted. In the software SNIU the hardware is controlled by the host's operating system software and not the SNIU's Scheduler module. Therefore, the software embodiment is not as trust worthy as the hardware one even though most of the software is identical. The Scheduler 50A is also utilized to control the flow of datagrams within the SNIU. The scheduler 50A provides timing for incoming datagrams and temporarily stores these datagrams in a buffer if earlier ones are being processed.

The Message Parser 50B is the first module in the TCB which processes an IP datagram received from the host computer. The Message Parser 50B checks the Association Table 76 and determines whether or not an association already exists for sending the datagram to its destination. If no association exists, the datagram is stored on the Waiting Queue and the Association Manager 54 is called to establish an association between this SNIU and the SNIU closest to the destination host. If an association does exist, the Session Manager 48 is called to encrypt the datagram, check security rules, and send the encrypted Protected User Datagram (PUD) to the peer SNIU.

When the Association Manager **54** is called, it prepares two messages to initiate the association establishment process. The first message is an Association Request Message which contains the originating host computer level and this SNIU's certificate (containing it's public signature key). This message is passed to the Fortezza API **56** which controls the Fortezza card which signs the message with this SNIU's private signature key. The second message is an ICMP Echo Request message which will be returned to this SNIU if it is received by the destination host. Both messages are passes to the network-side Host/Network Interface Module **46** to be transmitted to the destination host.

When a SNIU receives messages, the messages are first processed by the SNIU's receiving port's Host/Network Interface **46** which reassembles the messages and passes them to the trusted software. The Message Parser module **50B** passes the Association Request Message to the Association Manager **54** module and deletes the ICMP Echo Request message. The Association Manager **54** passes the message to the Fortezza API **56** which verifies the digital signature. If not valid, the Audit Manager **52** is called to generate an Audit Event Message to log the error. If the signature is OK, the Association Manager **54** saves a copy of the received Association Request Message in the Waiting Queue, adds this SNIU's certificate to the message, calls the Fortezza API **56** to sign the message, generates a new ICMP Echo Request message, and passes both messages to the Host/Network Interface module **46** to transmit the messages to the destination host. If the messages are received by any other SNIU's before reaching the destination host, this process is repeated by each SNIU.

If the destination host computer does not contain the Companion SNIU software, the host's communications protocol stack software automatically converts the ICMP Echo Request message to an ICMP Echo Reply and returns it to the SNIU which sent it. However, the destination host does not contain any software which can process the Association Request Message; so it is ignored (i.e., deleted).

If the destination host computer does contain Companion SNIU software, the host's data link layer software converts the stream of bits from the physical layer into packets which are passed to the Companion's Host/Network Interface module **46**. The hardware address headers are stripped off of the packets and saved; and the packets are re-assembled into IP datagrams which are passed to the Message Parser **50B**. The ICMP Echo Request message is ignored; and the Association Request Message is passed to the Fortezza API **56** to have the signature verified. If valid, the message is passed to the Association Manager module **54** which saves the originating host and SNIU data and generates an Association Grant Message. This message contains the SNIU's IP address (which is the same as the destination host's), the SNIU's certificate, the host's security level, and sealer keys for the originating SNIU and the previous intermediate SNIU (if there was one). The sealer keys (a.k.a. Message Encryption Keys) are explained elsewhere.

The Fortezza API **56** is then called to sign the message which is passed to the Host/Network Interface module **46**. The Association Grant Message is converted from an IP datagram to network packets and passed back to the host's hardware packet drivers (n the data link layer) for transmission back to the originating host.

Any intermediate SNIU's which receive the Association Grant Message process the message up through the communications stack protocol layers to the network layer which calls the Message Parser **50B** to process the message. The signature on the message is verified by the Fortezza API **56** and audited via the Audit Manager **52** if not valid. Otherwise, the validated message is processed by the Association Manager **54** module which removes and saves one of the sealer keys (a.k.a. a release key) which will be used by this SNIU and the previous SNIU (which generated the key) to authenticate PUD messages exchanged via this association in the future. The Fortezza API **56** is called to generate and wrap another sealer key to be shared with the next SNIU in the association path. The new key and this SNIU's certificate are appended to the message. The Fortezza API **56** aligns the message. The Host/Network Interface **46** transmits the message on its way back to the originating SNIU.

The originating SNIU re-assembles the Association Grant Message via the physical, data link **70** and network layers **68** as previously described. The signature is validated and audited if necessary. If valid, the Association Manager **56** uses the Fortezza API to unwrap the sealer key(s). If two keys are in the received message, the bottom key is a release key to be shared with the first intermediate SNIU; and the top key is an association key to be shared with the peer SNIU (which granted the association). If there is only one key, it is the association key which is shared with the peer SNIU; and the association path does not contain any intermediate SNIUs. Once the keys are stored and the Association Table **76** is updated, the association is established and the Session Manager **48** is called to transmit the original user datagram which was stored in the waiting Queue prior to issuing the Association Request Message.

The Session Manager **48** enforces the security policy, determines whether IP datagrams received from host computers can be transmitted via the network to their destination host, encapsulated these user datagrams in PUDs using the sealer keys for the appropriate association. The security policy is enforced by comparing the security levels of the host and destination. If the security levels are the same, the Session Manager checks the Association Table and identified the appropriate peer SNIU and sealer key(s). The user datagram is encrypted by the Fortezza API **56** using the association key. If the association contains any intermediate SNIUs, the Fortezza API **56** calculates a message authorization code using the release key. The Session Manager **48** creates a PUD addressed from this SNIU to the peer SNIU, encloses the encrypted user datagram, appends the message authorization code (if any), and passes the new datagram to the Host/Network Interface module **46** on the network-side of the SNIU. The datagram is broken into packets and transmitted as previously described.

If an intermediate SNIU receives the PUD, the data is passed through the data link layer software **70** to the network layer where the re-assembled datagram is passed to the Session Manager **48**. The source IP address is to identify the release key which is shared with the previous SNIU. The Fortezza API **56** uses the release key to verify the message authorization code. If not valid, the Session Manager **48** deletes the datagram and calls the Audit Manager **52** to generate an Audit Event Message. If the code is valid, it removes the code from the datagram, and uses the destination IP address to identify the release key shared with the next SNIU. The Fortezza API **56** generates a new message authorization code. The Session Manager **48** appends the new code and passes the datagram to the opposite port's Host Network Interface module.

When the peer SNIU (i.e., the destination IP address) received the PUD and it has been reassembled into a datagram, the Message Parser **50B** passes the datagram to the Session Manager **48**. The source IP address is used to identify the corresponding association key. The Fortezza

API 56 decrypts the original user datagram. The Session Manager checks the message authorization code and the security levels of the source and destination hosts. If the code is valid (i.e., the message was not modified during transmission over the network) and the security levels match, the decrypted datagram is passed to the Host/Network Interface 46 to be released to the destination host. If either is not correct, the Audit Manager 52 is called.

The following is a discussion of the protocol which applies to both the hardware and software SNIUs:

Address Resolution Messages

Address Resolution Protocol (ARP) allows a SNIU to find the hardware address of another SNIU or host on the same network, given its IP address. The SNIU broadcasts an ARP Request message which contains its hardware and IP addresses and the IP address of the target host. The target host (or other SNIU) returns to the requesting host an ARP Response message which contains the hardware address of the target host (or other SNIU).

Reverse Address Resolution Protocol (RARP) allows a SNIU which only knows its hardware address to obtain an IP address from the network. The host broadcasts a RARP Response which contains its hardware address, and a server on the network returns a RARP Response containing an IP address assigned to the requester's hardware address. All ARP and RARP messages have the same format and are contained within the frame data area of a single Ethernet frame (they are not IP datagrams). The format of the ARP and RARP messages is shown in the Table of FIG. 6.

Referring to FIG. 6, the HARDWARE TYPE is set to 0001 hex to indicate Ethernet. The PROTOCOL TYPE is set to 0800 hex to indicate IP addresses. The HLEN (hardware address length) is set to 06 hex bytes, while the PLEN (protocol address length) is set to 04 hex bytes. The OPERATION is set 0001 hex for an ARP request message, 0002 hex for ARP response message, 0003 hex for an RARP request message or 0004 hex for RARP response message. The SENDER'S HA contains the sender's 48 bit Ethernet hardware address, while the SENDER'S IP contains the sender's 32 bit IP address. The TARGET'S HA contains the target's 48 bit Ethernet hardware address, while the TARGET's IP contains the sender's 32 bit IP address.

When a SNIU broadcasts a request message, it fills in all of the data and the target's hardware address field is set to 000000 hex for an ARP message. If the message is a RARP, then the sender's and target's IP address fields are set to 0000 hex. When the target machine responds, it fills in the missing address and changes the operation field to indicate a response message. During an ARP, the target machine swaps the sender's and target's addresses so that the sender's address fields contains its addresses and the target's address fields contains the original requesting host's addresses. During a RARP, the server stores its addresses in the sender's address fields and returns the response to the original sender's hardware address.

When a SNIU receives a message, it performs the following processes:

ARP Request—If an ARP Request message is received on a SNIU's port A, the untrusted software in port A's memory segment determines if the sender's IP address is in port A's ARP cache. If not, it creates a new entry in the ARP cache and inserts the sender's hardware and IP addresses. Otherwise, the sender's hardware address is copied into the entry (overwriting any previous address) and packets (if any) waiting to be sent to the sender's IP address are transmitted. If the target's IP address is in port A's address list (i.e., a List of IP addresses which are accessed from port B), the untrusted software returns an ARP Response message swapping the SENDER'S and TARGET'S addresses and inserting port A's Ethernet hardware address into the SENDER's HA field. In either case, the untrusted software passes the ARP Request Trusted Computing Base (TCB).

The TCB checks port B's address list for the SENDER'S IP. If the SENDER'S IP is not in port B's address list, the TCB determines whether the SENDER'S IP is releasable to port B. If releasable, the TCB inserts SENDER'S IP into port B's address list. Secondly, the TCB determines whether a proxy ARP Request should be broadcast from port B. If an ARP Response message was not returned by port A and the target's IP address is not in port A's ARP cache, then the sender's IP is releasable to port B. Thus, causing the TCB to create a proxy ARP Request message. The TCB inserts port B's hardware and IP addresses in the SENDER'S address fields, copies the target's IP address from the original ARP Request into the TARGET'S IP field and then signals port B's untrusted software to broadcast the message.

Each time the TCB releases a proxy ARP Request, it creates an Anticipated Message in the form of a proxy ARP Response message. This proxy ARP Response message contains the original sender's addresses in the TARGETS fields, the target's IP address in the SENDER'S IP field and port A's hardware address is the SENDER'S HA field. This message is saved in the Anticipated Message list for port A and will be released to port A's untrusted software for transmission, if the anticipated ARP Response message is received on port B. Note that whether this message is released may involve the TCB modulating ARP Requests from a high network to a low network in order not to exceed the 100 bits per second covert channel bandwidth requirement.

ARP Response—If an ARP Response message is received on a SNIU's port A, the untrusted software in port A's memory segment determines if the sender's IP address is in port A's ARP cache. If not, it creates a new entry in the ARP cache and inserts the sender's hardware and IP addresses. Otherwise, the sender's hardware address is copied into the entry (overwriting any previous address) and packets (if any) waiting to be sent to the sender's IP address are transmitted. Finally, the untrusted software passes the ARP Response to the TCB.

The TCB checks port B's address list for the SENDER'S IP. If the SENDER'S IP is not in port B's address list, the TCB determines whether the SENDER'S IP is releasable to port B. If releasable, the TCB inserts it into port B's address list. Secondly, the TCB checks the Anticipated Message list for port B and determines whether the ARP Response was due to a proxy ARP Request made for a request originally received on port B. If the SENDER'S IP matches an entry in the Anticipated Message List and the message is releasable to port B, the TCB signals port B's untrusted software to create a proxy ARP Response message identical to the Anticipated Message and then removes the message from the Anticipated Message list for port B.

RARP Request—If a RARP Request message is received on a SNIU's port A, the untrusted software in port A's memory segment checks a flag to determine if the SNIU was initialized to act as a RARP server for the network attached to port A. If not, the received message is ignored. Otherwise, the untrusted software passes the RARP Request to the TCB.

The TCB determines whether the RARP Request can be released to port B. If releasable, it creates a proxy RARP Request message copying the TARGET'S HA from the received message and inserting port B's addresses in the SENDER'S HA and IP fields. Then the TCB passe the proxy

RARP Request message to port B's untrusted software for broadcast and creates an Anticipated message in the form of a proxy RARP Response message. The TCB copies the original TARGETS HA, inserts port A's hardware address in the SENDER'S HA and saves it in the Anticipated Message list for port A.

RARP Response—If a RARP Response message is received on a SNIU's port A, the untrusted software in port A's memory segment determines if the sender's IP address is in port A's ARP cache. If not, it creates a new entry in the ARP cache and inserts the sender's hardware and IP addresses. Otherwise, the sender's hardware address is copied into the entry (overwriting any previous address) and packets (if any) waiting to be sent to the sender's IP address are transmitted. Finally, the untrusted software inserts the TARGET'S IP into pen A's address list and passes the RARP Response to the TCB.

The TCB checks port B's address List for the SENDER'S IP. If the SENDER'S IP is not in pen B's address list, the TCB determines whether the SENDER'S IP is releasable to port B. If releasable, the TCB inserts it into port B's address list. Secondly, the TCB determines whether the TARGETS IP is releasable to port B. If releasable, the TCB creates a new entry in port B's ARP cache and inserts the TARGETS HA and IP. The TCB uses the TARGETS HA to find the appropriate proxy RARP Response message in port B's Anticipated Message List and copies the TARGETS IP and SENDER'S IP into the Anticipated message. Then the TCB signals port B's untrusted software to create a proxy RARP Response message identical to the Anticipated Message, and removes the message from the Anticipated Message list for port B.

Association Establishment Messages

SNIUs establish associations in order to authenticate each other, exchange security parameters, and establish trusted sessions for communication. The SNIUs utilize a standard ICMP Echo Request message to request an association and an ICMP Echo Reply message to grant an association.

When a host behind a SNIU attempts to communicate with someone else over the network, the SNIU stores the datagram from the host in a waiting queue and transmits an ICMP Echo Request to the intended destination. This message is used to identify other SNIU units in the communications path and to carry the originating SNIU's security parameters. The SNIU inserts the originating host's security level, appends its certificate and then signs the message. Each SNIU unit which receives this message authenticates the message, saves a copy of the previous SNIU's certificate, appends its certificate, and signs the message before sending it to the destination.

The destination host returns an ICMP Echo Reply to the originating SNIU. The first SNIU to receive this message is the terminating SNIU (i.e., closest to the destination) in the potential association's communications path. This SNIU determines if the association should be permitted (i.e. would not violate the security policy). If permitted, the SNIU grants the association, generates an encryption key for the association, and encrypts the key using the originating SNIU's public key (from its certificate). If the Echo Request message contained an intermediate SNIU's certificate, the SNIU also generates a release key and encrypts it using the intermediate SNIU's public key. In either case, the SNIU removes its and the originating SNIU's security parameters and signatures from the ICMP Echo Reply, stores the encrypted key(s), inserts the destination host's security level, appends its certificate, signs the message and sends it onto the originating SNIU.

Each intermediate SNIU (if any exist) which receives the Echo Reply message authenticates the previous SNIU's signature, extracts the release key, generates a new release key for the next SNIU, encrypts the key using the public key (from the certificate saved from the Echo Request message) of the next SNIU, removes the previous intermediate SNIU's certificate and signature, appends its own certificate and signature, and sends the message on the return path. When the originating SNIU receives the ICMP Echo Reply, it authenticates the message and extracts the key(s).

Once the association is granted, the originating SNIU fetches the originating host's datagram from the waiting queue and prepares to send it to the terminating SNIU in the newly established association. The SNIU uses the association key to encrypt the datagram for privacy. The SNIU further stores the encrypted datagram and the encryption residue into a new datagram from the originating SNIU to the terminating SNIU. If the association contains intermediate SNIUs, the originating SNIU uses the release key to calculate a second encryption residue and appends it to the datagram. Finally, the SNIU transmits the protected user datagram to the peer SNIU in the association.

When the protected user datagram is received by an intermediate SNIU (if any in the path), the intermediate SNIU fetches the release key corresponding to the previous SNIU and uses the release key to validate the datagram. If valid, the SNIU removes the release key residue from the datagram and checks to determine whether there are more intermediate SNIUs in the path before reaching the terminating SNIU. If another intermediate SNIU exists, the release key corresponding to the next intermediate SNIU is used to calculate a new release residue which is appended to the datagram. In either case, the datagram is sent on its way out the opposite port from which it was received.

When the terminating SNIU receives the protected user datagram, it uses the association key corresponding to the originating SNIU to decrypt and validate the datagram. If the source and destination hosts are at the same security level (i.e., a write-equal situation), the decrypted datagram is sent out the opposite port to the destination host. If the source host has a lower security level than the destination (i.e., a write-up situation), the SNIU predicts the response from the destination and saves it before sending the decrypted datagram to the destination host.

If the source host has a higher security level than the destination (i.e., a write-down situation), the received datagram (i.e., a response to a previous datagram from the lower level host) was predicted by the SNIU which sent the protected datagram. Therefore, this SNIU is assured that the classification of the received datagram is dominated by the lower Level destination host, so that the datagram is released to the destination. If a SNIU receives a user datagram from a native host which would be a write-down to the destination host and no predicted datagram is found, the received datagram is erased and the attempted write-down is audited.

Message Processing Tables

There are three tables which are used to process incoming and out-going messages including the Association Table, the Symmetric Key Table(Sym_Key) and the Certificate Table. Each SNIU has to Association tables, one for each port. Each entry contains data corresponding to a particular source or destination IP address. The Sym_Key table contains data corresponding to a particular message encryption key (MEK) which could be used as a release key or an association key. The Certificate table contains recently received certificates from other SNIU's.

Each table consists of a linked list of tokens in which the data for an entry in the table is stored in a token. The tokens

for each table have a unique data structure and are linked together in 'free' lists during initialization. When a new entry is made in one of the tables, a token is removed from the free list for that table's tokens, the data for the new entry is inserted in the appropriate fields of the token and the token is linked at the top of the table. When an entry is removed from a table, the 'previous' and 'next' tokens are linked, the data fields in the token are cleared and the token is linked at the bottom of the appropriate free list. Whenever the data in an entry is used, the token is removed from the table and relinked at the top of the table. In this way the oldest (i.e., least used) entry is at the bottom of the table.

If a new entry is needed and the free list is empty, the bottom token is removed from the table, the data fields are cleared, the new entry's data is inserted and the token is linked at the top of the table. In addition, when a SNIU removes the bottom (oldest unused) token in the Sym_Key Table, it also removes every token in the Association Table which pointed to the removed key. A SNIU does not send a Close Association Message when a certificate, key or Association Table entry is removed because many valid entries using the same association may still exist. The data structure for a token of the Association Table is shown in the Table of FIG. 7.

Referring to FIG. 7, NEXT is a pointer to the next token in the table or list. PREVIOUS is a pointer to the previous token in the table or list. IP ADDRESS is the IP address of the source/destination, while PEER SNIU IP ADDRESS is the address of the other terminating SNIU for the association. ASSOCIATION KEY POINTER points to the association MEK in the Sym_Key table. RELEASE KEY POINTER points to the release MEK in the Sym_Key table. The ASSOC-TYPE is set to 0001 hex for 'pending', 0002 hex for 'host' (i.e., the entry is for a host destination), 0003 hex for 'sniu' (i.e., the entry is for a SNIU destination), 0004 hex for 'native host' (i.e., no peer SNIU) or 0005 hex for 'audit catcher'. The RELKEY-TYPE is set to 0001 hex for 'in' (i.e., use to validate release key residue), 0002 hex for 'out' (i.e., use to add release key residue) or 0003 hex for 'both'. SECURITY-LEVEL indicates the security level of the source/destination, while SPARE is an unused byte to keep addressing on a 32-bit boundary.

Referring to FIG. 8, there is shown the data structure of a token for the Sym_Key Table according to the present invention. NEXT is a pointer to the next token in the table or list, while PREVIOUS is a pointer to the previous token in the table or list. DISTINGUISHED NAME is the 128 byte name in certificate from the other SNIU using this key. MEK is the 12 byte wrapped key (association or release) shared with the another SNIU. IV is the 24 byte initialization vector associated with the MEK. CERTIFICATE POINTER points to the other SNIU's certificate in the Certificate table. INDEX is a Fortezza card key register index which indicates if and where the key is loaded (1–9 are valid key register indexes, while 0 indicate that the key is not loaded on the Fortezza). SPARE is an unused byte to keep addressing on a 32-bit boundary.

Message Flag

Any message (IP datagram) which is generated or modified by a SNIU unit contains a Message Flag in the last four bytes of the datagram. The first byte is the message type field, the second byte is the message format field and the third and fourth bytes are the Flag. Note that all message types are signed except for a Protected User Datagram (PUD) which uses MEK residues for integrity and authentication.

Waiting Que and Schedule Table

The Waiting Que is used to store IP datagrams for potential future processing based on an anticipated event. For every entry made in the Waiting Que, a corresponding entry is made in the Schedule Table. The Schedule Table is used to automatically process entries in the Waiting Queue if they have not been processed within some predetermined amount of time (i.e. the anticipated event does not occur). The Schedule Table entry contains a time-out field (which is set to the current time plus some reasonable delta representing the maximum waiting period) and a function pointer (which indicates which subroutine should be called if time expires before the Waiting Queue entry is processed). The Schedule Table is checked in the main executive loop of the TCB, expired entries are removed and the corresponding datagrams in the Waiting Queue are processed by the designated subroutine.

For example, when a SNIU receives a user datagram from a native host which is destined for another host for which there is no existing association, the SNIU stores the user datagram in the Waiting Queue and transmits an Association Request message. When the Association Grant message is received, the user datagram is removed from the Waiting Queue, the corresponding Schedule Table entry is deleted, the user datagram is encrypted and sent to the peer SNIU of the association. If an Association Grant message is never received, the Schedule Table entry expires which calls a subroutine to delete the user datagram from the Waiting Queue.

Another example is when the SNIU sends an Audit Event message to an Audit Catcher. The transmitted datagram is stored in the Waiting Queue. When the Receipt message is received from the Audit Catcher, the original Audit Event datagram is removed from the Waiting Queue and the corresponding Schedule Table entry is deleted. If the Schedule Table entry expires, the designated subroutine is called which re-transmits the Audit Event message stored in the Waiting Queue and a new entry is made in the Schedule Table.

Generating and Exchanging MEKs

Message Encryption Keys (MEKs) are generated during the association establishment process (previously described) and are exchanged via the Association Grant Message. When a SNIU generates an MEK, it simultaneously generates an initialization vector (IV).

When a SNIU exchanges an MEK with another SNIU, it generates a random number, RA, which is required to encrypt (i.e., wrap) the MEK. The key exchange algorithm is designed so that only the sending and receiving SNIUs can decrypt the MEK and use it. The sender wraps the MEK for transmission using the destination's public Key RA.RB (which is always set=1) and the sender's private key. IVs which were generated with release keys are transmitted in the clear with the wrapped MEK in the Association Grant Message, while IVs which were generated with association keys are ignored. The recipient unwraps the key using its private key RA.RB, and the sending SNIU's public key. Once unwrapped, the safe exchange is complete.

Each SNIU re-wraps the MEK using its storage key (Ks), stores the MEK and the IV (if the MEK is a release key) in the Sym_Key Table, stores the pointer to the MEK in the Association Table and stores the DN (of the other SNIU sharing this MEK) in the Sym_Key Table entry.

Using MEKs and IVs

Message Encryption Keys (MEKS) are used as association and release keys to provide confidentiality, integrity and authentication of user datagrams during an association between two SNIUS. IVs are used to initialize the feedback

loop in the Skipjack encryption algorithm for most modes of operation. Encrypting identical data using the same MEK, but different IVs, will produce different cipher text. In fact, the Fortezza card requires the user to generate a new IV for each encryption event in order to assure that each message looks different when encrypted.

When a SNIU encrypts a user datagram it first generates a new IV for the association key, encrypts the datagram, appends the encryption residue for integrity and authentication purposes, and appends the new IV. If the association involves intermediate SNIUs, the SNIU performs a decrypt operation on the newly encrypted datagram, residue and IV. The decrypt operation uses the release key and release Key IV. The release key IV is never changed since the encrypted data is always guaranteed to be unique even if the original datagram is not. The release key residue is appended to the protected user datagram. The completed protected user datagram is then transmitted.

Received Message Processing

When a SNIU receives an IP datagram, it checks the destination address in the header and determines if it is the intended recipient. Then, the SNIU checks the last four bytes of the IP datagram for the Message flag and determines the type and format of the received message.

Destination SNIU Message Processing

When a SNIU receives an IP datagram which is addressed to it, the message should be one of the following messages: an Audit Event, Audit Catcher list, Audit Mask, Association Close, Association Request, Association Grant, Association Denial, Association Unknown, Protected User Datagram, Receipt and Certificate Revocation List. If it is not, the SNIU audits the event. The only exceptions are ICMP datagrams which are processed by the receiving port's untrusted software and not passed to the trusted computing base.

Audit Event—If the SNIU is not configured to be an Audit Catcher, it will audit the event sending the source IP address of the received message to its primary Audit catcher.

If the SNIU is configured to be an Audit Catcher, it verifies the signature on the message, increments its received audit event sequence number, generates a time stamp, and prints the sequence number, time stamp, source IP address, and ASCII character string from the message. Once the event has been recorded, the Audit catcher SNIU generates a Receipt Message (copies the audit event counter from the received message and inserts it in the message number field), sends it, and checks the receiving port's Association Table for an entry for the source of the received message. If an entry does not exist (i.e., this was the first communication from the source SNIU), the Audit Catcher makes an entry, marks the association type as 'sniu', and sends the SNIU the current audit mask.

Audit Catcher List—The SNIU verifies the signature on the message, stores the new list of Audit catchers in the Configuration table, generates a receipt message, and audits the event.

Audit Mask—the SNIU verifies the signature on the message, stores the new audit mask in the Configuration Table, generates a Receipt Message, and audits the event (in case someone else other than the Audit Catcher is distributing new audit masks). In addition, if the receiving SNIU is an Audit Catcher, it distributes the new audit mask to every destination in the Association Table with an association type of 'sniu'.

Association Close—When a SNIU receives a valid protected User Datagram, but cannot find the destination's Association Table entry, it sends an Association close message back to the originating SNIU and audits the event. The originating SNIU verifies the signature on the received Association Close Message, extracts the original destination host's IP, removes the host's entry from its Association Table and audits the event. It does not remove the peer SNIU's entry nor entries from the Sym_Key table as they might be supporting other associations.

Association Request—This message can only be received by a SNIU which originally transmitted it as an ICMP echo request to some unknown destination which converted it to an Echo reply and returned it to the originator without encountering another SNIU. Therefore, the SNIU uses the source IP address to find the destination's entry in the Association Table, changes the association type from 'pending' to 'native host', sets the security level to that port's security level, finds the original host's user datagram in the Waiting Queue, removes the corresponding entry from the Schedule table, and compares the source and destination security levels to determine if the user program can be sent to the destination. If the comparison indicates a write-up situation, the SNIU generates and saves an anticipated message and releases the original datagram to the destination port. If a write down situation, the SNIU determines if the data gram was predicted and sends the anticipated message or audits as previously described. If a write equal, the datagram is released to the destination port. This procedure is repeated for each entry in the Waiting Queue which is intended for the same destination.

Association Grant—The SNIU verifies the signature in the datagram and updates the receiving port's Association Table entries for the host destination and peer SNIU. The SNIU finds the entry for the destination host, changes the association type from 'pending' to 'host', copies the peer SNIU's IP, extracts and unwraps the association MEK (and release MEK if needed), stores the re-wrapped key(s) in the Sym_Key table, marks the release key type as 'out' (if a release key exists), copies the destination host's security level, and determines if an entry exists for the peer SNIU. If not, the SNIU creates a new entry for the peer SNIU, copies the association and release key pointers and release key type from the destination host's entry, and marks the association type as 'sniu'.

Once receiving the port's Association Table has been updated, the SNIU finds the original hosts user datagram in the waiting Queue, removes the corresponding entry from the Schedule table, and compares the source and destination security levels to determine if the user datagram can be sent to the destination. If the source's security level is dominated by (i.e., less than or equal to) the destination's security level, the SNIU creates a Protected user Datagram (PUD). The SNIU sets the destination to the peer SNIU's IP, sets the protocol type to indicate a SNIU Message, uses the association key to encrypt the entire received datagram, inserts the ciphertext and IV, appends the association residue, generates and inserts a release residue (if the destination host's Association Table entry contains a pointer to a release key), appends the appropriate SNIU Message Flag, and sends the datagram. If the source host is not dominated by the destination (i.e., potential write down, the attempted write down is audited. This procedure is repeated for each entry in the Waiting Queue which is intended for the same destination.

Association Unknown—A SNIU sends an Association Unknown message (and generates audit notices) when a protected user datagram is received and a corresponding Association Table entry does not exist. The message is sent back to the source SNIU and contains the destination SNIU's IP address. When a SNIU receives an Association

Unknown Message, it deletes every entry in the Association Table in which the peer SNIU address matches the returned destination SNIU IP. Subsequent user datagrams from the same host sent to the same destination will initiate an Association Request to re-establish the association. Any SNIU involved in establishing an association for which it already has keys (association and/or release keys) will suggest the same key as originally used.

Protected User Datagram—the SNIU uses the source IP to find the appropriate entry in the receiving port's Association Table and retrieve the association key to decrypt and validate the received datagram. If the decryption residue does not match, the even is audited. otherwise, the SNIU uses the destination host's IP to find the appropriate entry in the opposite port's Association Table, retrieves the destination host's security level, and compares it to the security level in the received datagram. If a write-up situation, the SNIU generates an anticipated message. However, regardless of the relative security levels, the decrypted and validated user datagram is sent to the destination host.

If a terminating SNIU receives a PUD and validates the residue but cannot deliver the user datagram because it cannot find the destination host in the Association Table, then the SNIU returns an Association close message to the originating SNIU (containing the destination host's IP) and audits the event.

Receipt—A Receipt message is sent by an Audit catcher to a SNIU for Audit Catcher Request and Audit Event messages. The SNIU uses the message number in the received datagram to locate the saved copy of the original message in the Waiting Queue and remove it and the corresponding Schedule Table entry. If the original message was an Audit Catcher Request Message, the SNIU locates the Association Table entry for the Audit catcher and changes the association type from 'pending' to 'audit catcher'. If time expires in the Schedule Table entry before the Receipt Message is received, the SNIU will retransmit the original message. If no receipt is received after TBD attempts, the SNIU will switch to the next Audit Catcher in the list. If all Audit Catchers are attempted without success, the SNIU will check a configuration parameter to determine whether to continue without audit or halt.

SNIUs issue Receipt messages to Audit catchers for Audit Catcher List, Audit Mask, and Certificate Revocation List messages. When an Audit Catcher receives a receipt, it uses the returned message number to remove the copy of the message from the Waiting Queue and the corresponding Schedule table entry.

Certificate Revocation List—if a Certificate revocation List (CRL) is received, the SNIU returns a receipt to the source and checks the Sym_Key Table for any keys which were received from (or sent to) another SNIU with a revoked certificate, the SNIU deletes the certificate from the Certificate Table (if it is still there), deletes the Sym_Key Table entry, and deletes every entry in the Association Table which pointed to the key. Note that deleting a table entry means to unlink the token from the table, clear the token's memory, and re-link the token in the token's free list.

Non-Destination SNIU Message Processing

When a SNIU receives an IP datagram which is not addressed to it, the message should be one of the following types of Dragonfly formatted messages. If it is not, the SNIU will assume the IP datagram is from a native host.

Audit Event—The SNIU verifies the signature on the message and releases the message out the opposite pen

Audit Catcher List—The SNIU verifies the signature on the message and releases the message out the opposite port.

Audit Mask—The SNIU verifies the signature on the message and releases the message out the opposite port.

Association Close—The SNIU verifies the signature on the message and releases the message out the opposite port.

Association Request—When a SNIU receives an Association Request it first checks the IP header to determine if the datagram is an ICMP Echo Request or an ICMP Echo Reply.

If it is an ICMP Echo Request, the SNIU validates the signature at the bottom of the message and checks the receiving port's Association Table for an entry with the originating SNIU's IP address. If the receiving SNIU cannot find an entry, it creates one, marks the association type as 'pending', stores the previous SNIU's certificate in the Certificate Table (if it wasn't already there), updates the Sym-Key Table entry for the Distinguished Name (DN), and stores the pointer to the Sym_Key Table entry in the release key pointer field in the Association Table entry. If the previous SNIU was an intermediate SNIU (i.e., the Message Format field of the SNIU Message Flag is 'Signed Type 2'), this SNIU marks the release key type field as 'out' and removes the previous SNIU's certificate and signature. In either case, this SNIU appends its certificate and signature and sends the message out the other port. It does not make any entry in the out-going port's Association Table.

If it is an ICMP Echo Reply, this SNIU is the terminating SNIU which must generate the Association Grant Message. Before the SNIU can validate the received message, it must reconstruct the message to match the form it was in when the SNIU signed it before the destination host converted the ICMP Echo Request into an ICMP Echo Reply. Therefore, the SNIU exchanges the source and destination IP addresses in the datagram header and changes the type field in the ICMP header from a request (8) to a reply (0). Then the SNIU validates the signature at the bottom of the newly reconstructed message using its own public key. If the signature cannot be validated, the event is audited.

If the ICMP Echo Reply is valid, the SNIU creates or updates three Association Table entries. First, it creates an entry (if it doesn't already exist) in the receiving port's Association Table for the original destination host (using the destination IP from the modified datagram header), marks the association type as 'native host' and stores the receiving port's security level in the security level field. Second, it updates the entry in the opposite port's Association Table for the peer SNIU (using the source IP from the modified datagram header). If the release key type is marked 'out' or 'both', then the association path contains at least one intermediate SNIU; therefore, the SNIU extracts the peer SNIU's certificate from the datagram, stores it in the Certificate Table, stores the pointer to the certificate and the DN in a Sym_Key Table entry, and stores the pointer to the Sym_Key Table entry in the association key pointer field of the Association Table entry. If there aren't any intermediate SNIUs, the pointer in the release key pointer field is copied to the association key pointer field; and the release key pointer field is cleared. In either case the association type is marked as 'sniu'. The third Association Table entry is for the originating host. It's IP and security level are in the data portion of the received datagram. The security level is copied into the entry, the association type is marked as 'host', and the rest of the data is copied from the peer SNIU entry. Finally, the SNIU generates the association key (and if necessary, the release key) and stores the key(s) in the Sym_Key Table entry(s).

Once the Association Table entries are updated, an Association Grant Message is generated. The SNIU uses the peer

(i.e., originating) SNIU's IP for the destination, uses the original destination host's IP for the source, and marks the protocol and type fields to indicate an ICMP Echo Reply. The SNIU inserts its IP address, its certificate, its host's security level, the association key data (wrapped key and RA), and if necessary, the release key data (the wrapped key, RA and IV). The SNIU Message Flag is inserted at the bottom marking the type as Association Grant and the format as Signed Type 1 to indicate only one certificate. The message is signed and sent.

Association Grant—The SNIU validates the signature at the bottom of the received datagram and, if not correct, audits the event. Otherwise, since it is not the destination, the SNIU is an intermediate SNIU somewhere in the path between the two peer SNIUs. The SNIU creates an entry (if one doesn't already exist) in the receiving port's Association Table for the IP of the terminating SNIU which granted the association (note that the terminating SNIU's IP is not in the header of the received datagram, rather it is in the data area), marks the association type as 'sniu', marks the release key type as 'in' (if the format is 'Signed Type 1') or 'both' (if the format is 'Signed Type 2'), extracts the release key data (i.e., the wrapped MEK, RA and IV), unwraps and stores the release key in the Sym_Key Table, stores the release key IV in the same Sym_Key Table entry, stores the pointer to the release key in the Association Table, stores the certificate in the Certificate Table, and stores the pointer to the certificate and the DN in the Sym_Key Table entry.

Next, the SNIU uses the destination IP address in the header of the received Association Grant Message to find the destination's entry in the opposite port's Association Table. If the association type is 'pending', the SNIU uses the release key pointer to fetch the saved certificate of the next SNIU, generates release key data (an MEK, RA, and IV), stores the wrapped MEK and IV in the Sym_Key Table entry, and changes the association type to 'sniu'. If the association type is 'NULL', the SNIU changes it to 'in'; otherwise, it is marked as 'both'.

Finally, the SNIU rebuilds the Association Grant Message to send on to the destination. The SNIU copies the received datagram up to and including the association key data and the certificate of the SNIU which originated the Association Grant Message, inserts its certificate and the release key data, and signs and sends the datagram.

Association Unknown—The SNIU verifies the signature on the message and releases the message out the opposite port.

Protected User Datagram—The SNIU uses the source IP address to find the appropriate entry in the receiving port's Association Table, fetches the release key, and verifies the release key residue. If the release residue is not correct the datagram is deleted and the event audited. Otherwise, the SNIU uses the destination IP address to find the appropriate entry in the opposite port's Association Table, fetches the release key, generates the new release residue, overwrites the old release residue, and sends the datagram on to the destination.

Receipt—The SNIU verifies the signature of the message and releases the message out the opposite port.

Certificate Revocation List—The SNIU verifies the signature on the message and releases the message out the opposite port.

Native Host Message—When a SNIU receives a user datagram from a native host, the SNIU creates an entry (if one doesn't already exist) in the receiving port's Association Table for the source host's IP, marks the association type as 'native host', sets the security level to the receiving port's security level, and checks the opposite port's Association Table for the destination's IP address.

If an entry does not already exist for the destination, the SNIU creates a new entry, marks the association type as 'pending', stores the received datagram in the Waiting Queue, makes a corresponding entry in the Schedule Table, creates an Association Request Message and sends it.

If an Association Table entry exists for the destination and the association type is 'pending', the SNIU stores the received datagram in the Waiting Queue, linking it to other datagrams for the same destination.

If an Association Table entry exists for the destination and the association type is 'host', the SNIU compares the source host's security level to the destination host's security level. If the source's security level is dominated by (i.e., less than or equal to) the destination, the SNIU creates a Protected User Datagram (PUD). The SNIU sets the destination to the peer SNIU's IP, sets the protocol type to indicate a SNIU Message, uses the association key to encrypt the entire received datagram, inserts the ciphertext and IV, appends the association residue, generates and inserts a release residue (if the Association Table entry contains a pointer to a release Key), appends the appropriate Dragonfly Message Flag, and sends the datagram. If the source host is not dominated by the destination (i.e., a potential write down), the SNIU determines if this datagram was anticipated. If a matching datagram was predicted the anticipated datagram is transformed into a PUD (as described above) and sent. If an anticipated message is not found, the attempted write-down is audited.

If an Association Table entry exists for the destination and the association type is any other bona fide type, i.e., 'native host', 'sniu', or 'audit catcher', the SNIU compares the source and destination port's security levels to determine if the datagram can be allowed to proceed. If the comparison indicates a write-up situation, the SNIU generates and saves an anticipated message and releases the original datagram to the destination port. If a write-down situation, the SNIU determines if the datagram was predicted and sends the anticipated message or audits as previously described. If a write-equal, the datagram is released to the destination port.

It is to be will be understood that the embodiments described herein are merely exemplary of the principles of the invention, and that a person skilled in the art may make many variations and modifications without departing from the spirit and scope of the invention. All such variations and modifications are intended to be included within the scope of the invention as defined in the appended claims.

What is claimed is:

1. A multi-level security device for providing security between a user and at least one computer network, wherein the user is selected from the group consisting of a host computer and at least a second network, comprising:

a secure network interface Unit (SNIU) that operates at a user layer communications protocol, said SNIU communicates with other like SNIU devices by establishing an association at a session layer of a communication stack in order to provide secure end-to-end communications, comprising:

a host/network interface for receiving messages sent between said user and said at least one network, said interface operative to convert said received messages to and from a format utilized by said at least one network;

a message parser for receiving said messages from said host/network interface, determining whether said association already exists with another SNIU device and providing a signal indicative of said determination;

a session manager coupled to said interface for identifying and verifying said user requesting access to said network, said session manager also responsive to said signal from said message parser for transmitting said messages received from said user when said message parser determines said association already exists; and

an association manager coupled to said interface and responsive to said signal from said message parser for establishing an association with other like SNIU devices when said message parser determines said association does not exist, wherein said message parser stores said messages in a wait queue until said association is established.

2. The device of claim 1, wherein said association manager generates an ICMP Echo request message in order to establish an association with other like SNIU devices.

3. The device of claim 2, wherein said ICMP Echo request includes the user's security level.

4. The device of claim 1, wherein said association manager generates a ICMP Echo reply in order to grant an association with other like SNIU devices.

5. The device of claim 1, which further includes a card reader configured to perform integrity and authenticating functions.

6. The device of claim 1, wherein said session manager protects the security communications between said computer device and said network by implementing a security policy selected from a group consisting of discretionary access control, mandatory access control, object reuse, labeling, denial of service detection, data type integrity, cascading control and covert channel use detection.

7. The device of claim 1, wherein said SNIU further includes means for performing a defined trusted session layer protocol (TSP), said TSP constituting said user layer communications protocol.

8. The device of claim 1, wherein said SNIUs during said association perform a function selected from the group consisting of authenticating each other, exchanging security parameters and establishing trusted sessions for further communications.

9. The device of claim 1, wherein said host/network interface also processes Address Resolution Protocol (ARP) and Reverse Address Resolution Protocol (RARP) messages.

10. The device of claim 1, wherein said SNIU further includes a scheduler coupled between said host/network and said message parser for controlling the flow of said data within said SNIU.

11. The device of claim 1, wherein said SNIU further includes an audit manager coupled to said association manager for generating audit event messages when a message is received with an invalid authorization code.

12. A method of providing a multi-level network security system for a user and at least one computer network, wherein said user is selected from the group consisting of a host computer and at least a second network, comprising:

placing a secure network interface Unit (SNIU) that operates at a user layer communications protocol, said SNIU communicates with other like SNIU devices by establishing an association at a session layer of a communication stack in order to provide secure end-to-end communications, said SNIU performing a plurality of security functions including:

receiving messages sent between said user and said at least one network;

converting said received messages to and from a format utilized by said at least one network;

identifying and verifying said user requesting access to said network;

determining whether said association already exists with another SNIU device;

transmitting said messages received from said user when said association already exists;

temporarily storing said received messages from said computer device when said association does not exist; and

establishing an association with other like SNIU devices when said association does not exist.

13. The method of claim 12, which further includes encrypting outgoing messages and decrypting incoming messages of said SNIU.

14. The method of claim 13, which further includes generating and writing cryptographic residues for outgoing messages and validating cryptographic residues for incoming messages.

15. The method of claim 12, wherein said SNIU protects the security communications between said computer device and said network by implementing a security policy selected from a group consisting of discretionary access control, mandatory access control, object reuse, labeling, denial of service detection, data type integrity, cascading control and covert channel use detection.

16. The method of claim 12, which further includes generating an audit trail.

17. The method of claim 12, wherein said association is established by said SNIU:

storing a datagram including information received from the user;

transmitting an ICMP Echo request message in order to request said association with other like SNIU devices;

waiting to receive an ICMP Echo reply message from other like SNIU devices in order to grant said association;

retrieving said stored datagram when said SNIU receives said ICMP Echo reply message; and

transmitting said retrieved datagram.

18. The method of claim 17, wherein said ICMP Echo request includes the user's security level.

19. The method of claim 12, which further includes transmitting an ARP Request message including said SNIU's hardware and IP address, and IP address of other like SNIU devices in order to locate other like SNIU devices on the network.

* * * * *

US007336615B1

(12) **United States Patent**
Pan et al.

(10) **Patent No.: US 7,336,615 B1**
(45) **Date of Patent: Feb. 26, 2008**

(54) **DETECTING DATA PLANE LIVELINES IN CONNECTIONS SUCH AS LABEL-SWITCHED PATHS**

(75) Inventors: **Ping Pan**, Emerson, NJ (US); **Nischal Sheth**, Sunnyvale, CA (US)

(73) Assignee: **Juniper Networks, Inc.**, Sunnyvale, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1091 days.

(21) Appl. No.: **10/179,927**

(22) Filed: **Jun. 25, 2002**

Related U.S. Application Data

(60) Provisional application No. 60/301,050, filed on Jun. 25, 2001.

(51) **Int. Cl.**
H04L 12/26 (2006.01)

(52) **U.S. Cl.** **370/248**; 370/225

(58) **Field of Classification Search** 370/216–228, 370/241–253, 395.3, 401, 229–240
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,775,239 B1 * 8/2004 Akita et al. 370/248
6,895,441 B1 * 5/2005 Shabtay et al. 709/238
2002/0054405 A1 * 5/2002 Guo et al. 359/118
2005/0281192 A1 * 12/2005 Nadeau et al. 370/217
2006/0013142 A1 * 1/2006 Hongal et al. 370/248

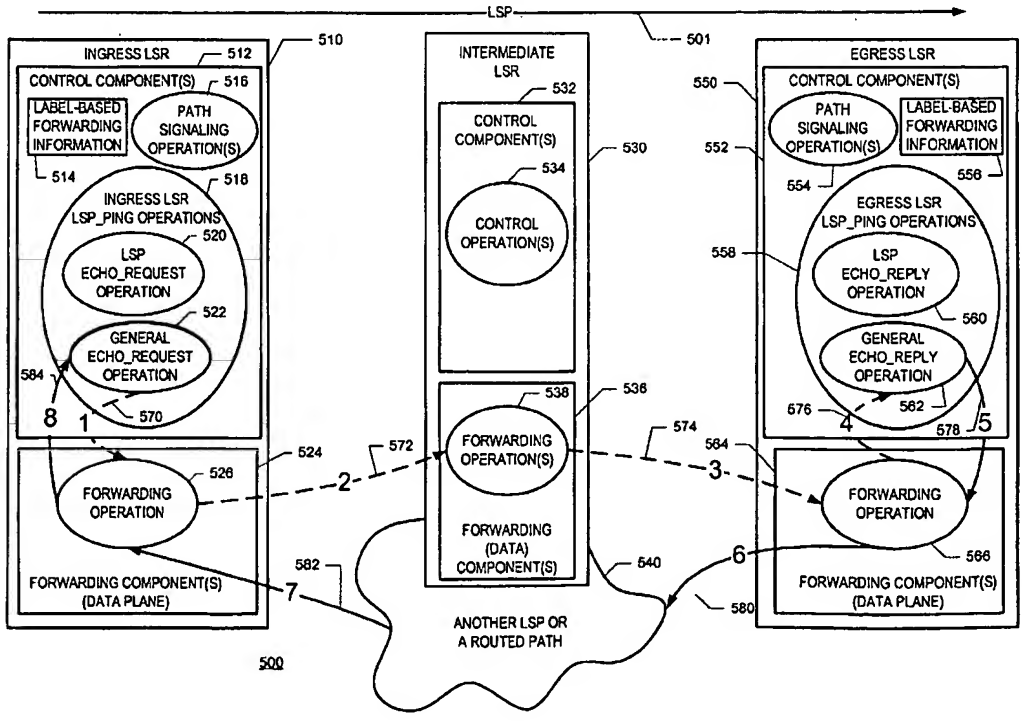
* cited by examiner

Primary Examiner—Doris H. To
Assistant Examiner—Feben Micael Haile
(74) *Attorney, Agent, or Firm*—Straub and Pokotylo; John C. Pokotylo

(57) **ABSTRACT**

Testing the liveliness of a data plane of a label switched path (LSP) using a two stage approach. The first stage may use a general echo request operation that may be implemented using hardware. Therefore, the first stage does not heavily burden the control plane of the LSR. If a suspect LSP passes the first stage of the diagnostic operation, nothing more needs to be done. If, however, the suspect LSP fails the first stage, the diagnostic operation proceeds to a second stage. The second stage of the diagnostic operation sends probing massages through the suspect LSP, but uses the control plane to deliver the acknowledging messages. If the suspect LSP fails the second stage of the diagnostic operation, the ingress LSR can infer that the LSP is down, and begin recovery actions. The probing messages may include padding so that MTU limits can be tested. In addition, the probing messages may be encapsulated in a protocol that allows flow control, thereby protecting an LSR that can receive such messages from DoS attacks.

47 Claims, 11 Drawing Sheets



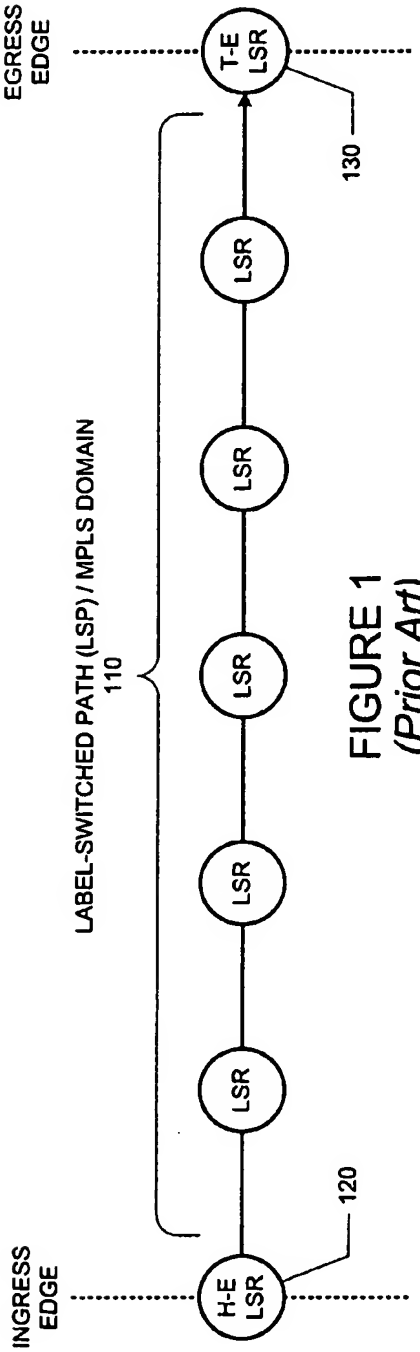


FIGURE 1
(Prior Art)

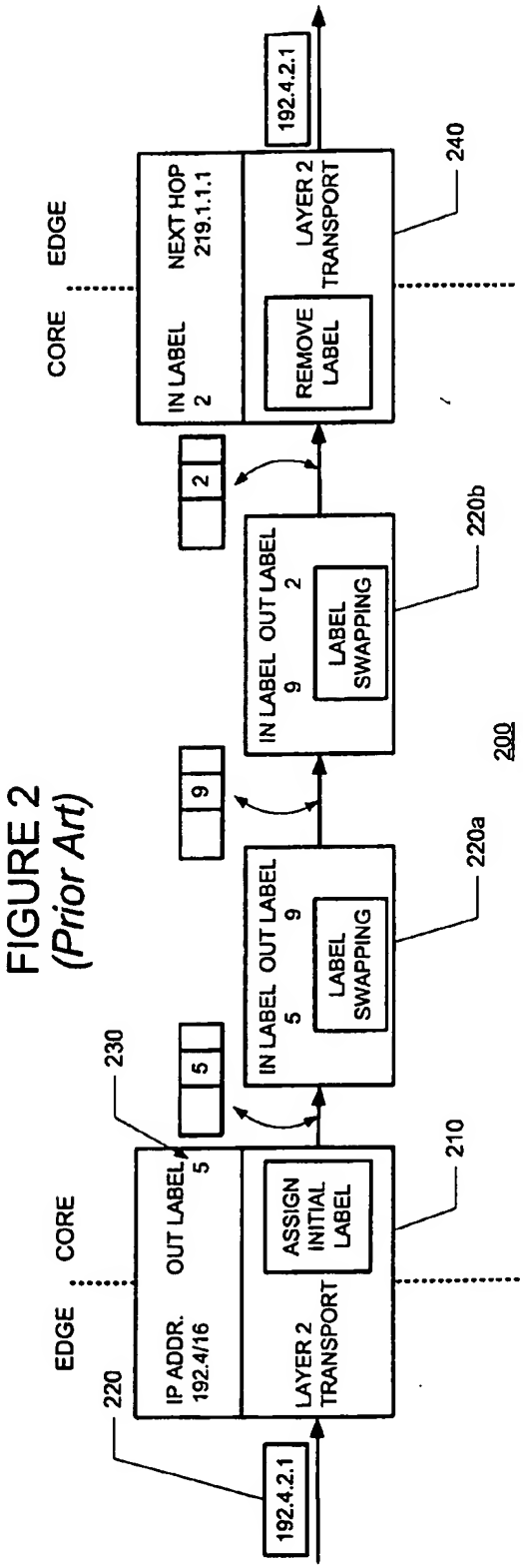


FIGURE 2
(Prior Art)

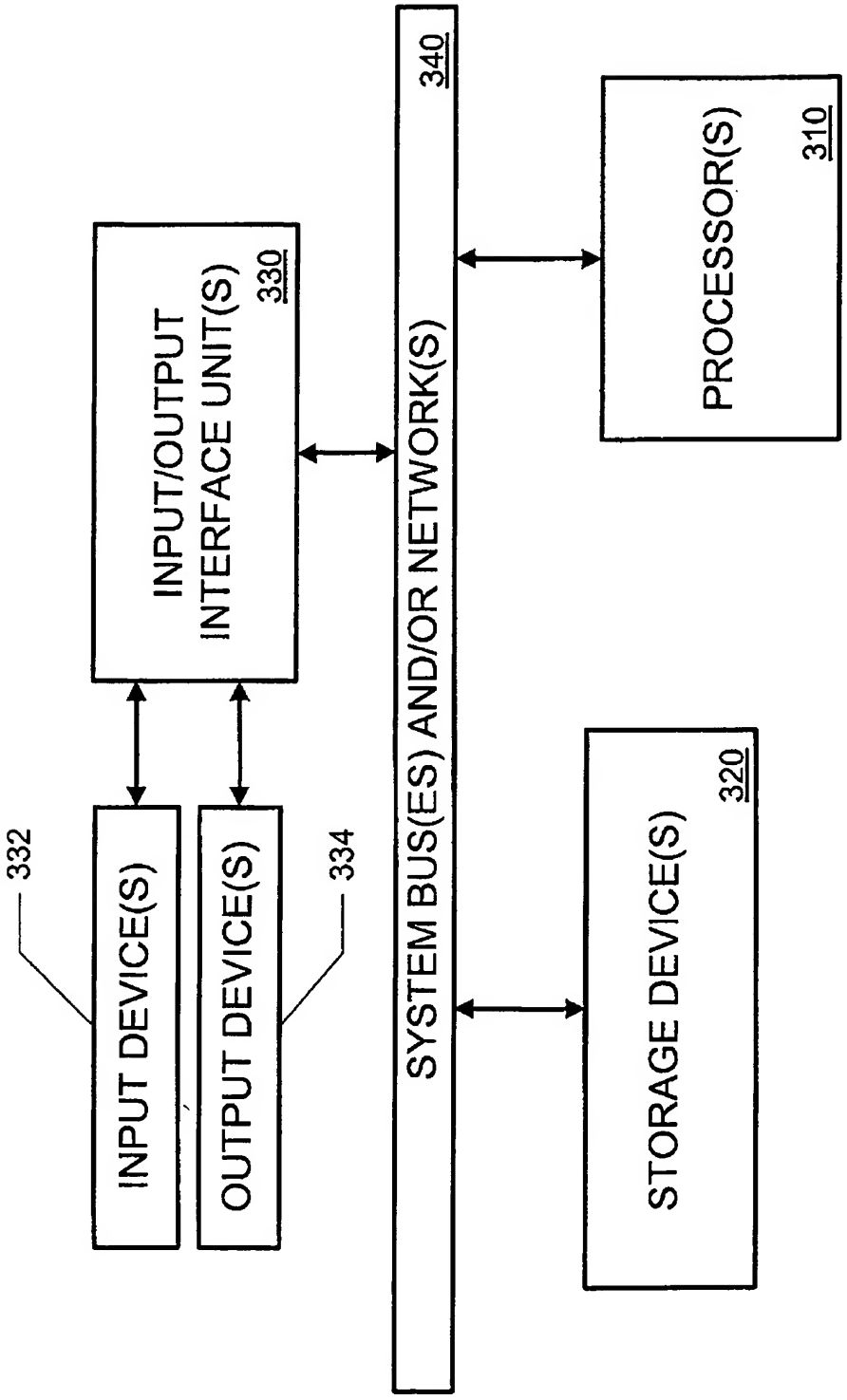

INPUT/OUTPUT INTERFACE UNIT(S)
330
PROCESSOR(S)
310
SYSTEM BUS(ES) AND/OR NETWORK(S)
340
INPUT DEVICE(S)
332
OUTPUT DEVICE(S)
334
STORAGE DEVICE(S)
320
300

FIGURE 3

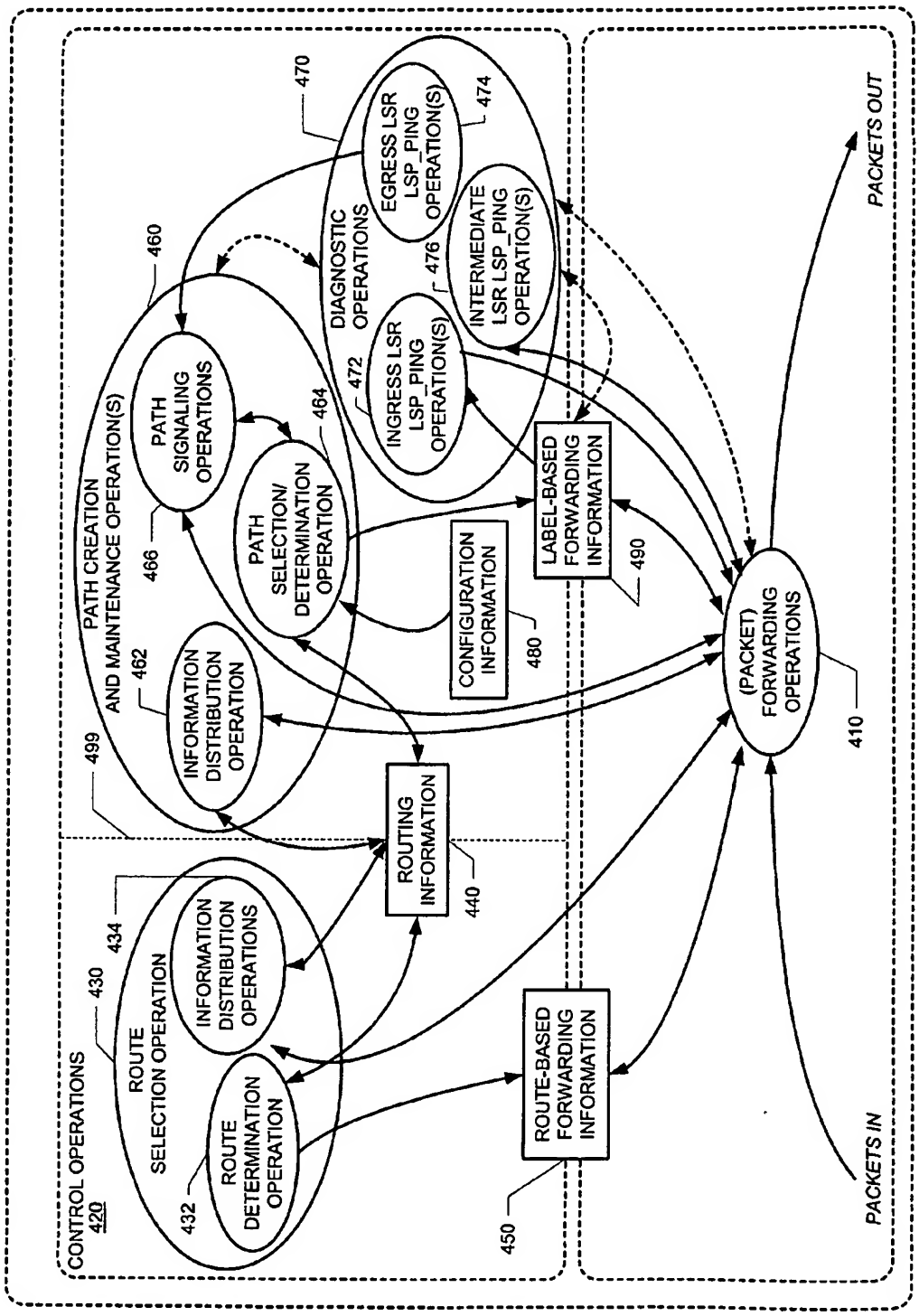

CONTROL OPERATIONS
420
ROUTE SELECTION OPERATION
430
ROUTE DETERMINATION OPERATION
432
INFORMATION DISTRIBUTION OPERATIONS
434
ROUTE-BASED FORWARDING INFORMATION
450
ROUTING INFORMATION
440
PATH CREATION AND MAINTENANCE OPERATION(S)
460
PATH SIGNALING OPERATIONS
466
PATH SELECTION/ DETERMINATION OPERATION
464
INFORMATION DISTRIBUTION OPERATION
462
499
CONFIGURATION INFORMATION
480
DIAGNOSTIC OPERATIONS
470
INGRESS LSR LSP_PING OPERATION(S)
472
EGRESS LSR LSP_PING OPERATION(S)
474
INTERMEDIATE LSR LSP_PING OPERATION(S)
476
LABEL-BASED FORWARDING INFORMATION
490
(PACKET) FORWARDING OPERATIONS
410
PACKETS IN
PACKETS OUT
400

FIGURE 4

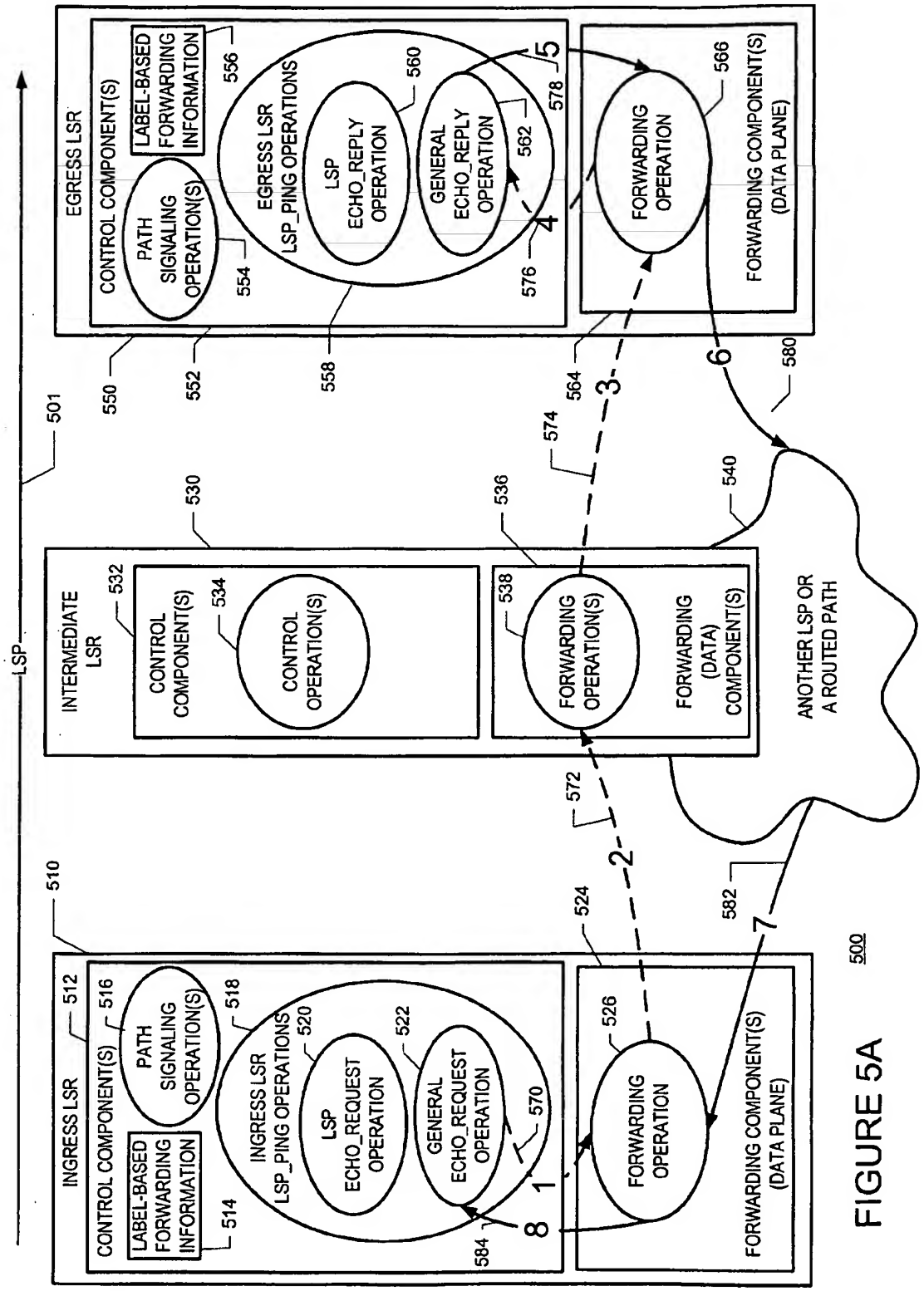


FIGURE 5A 500

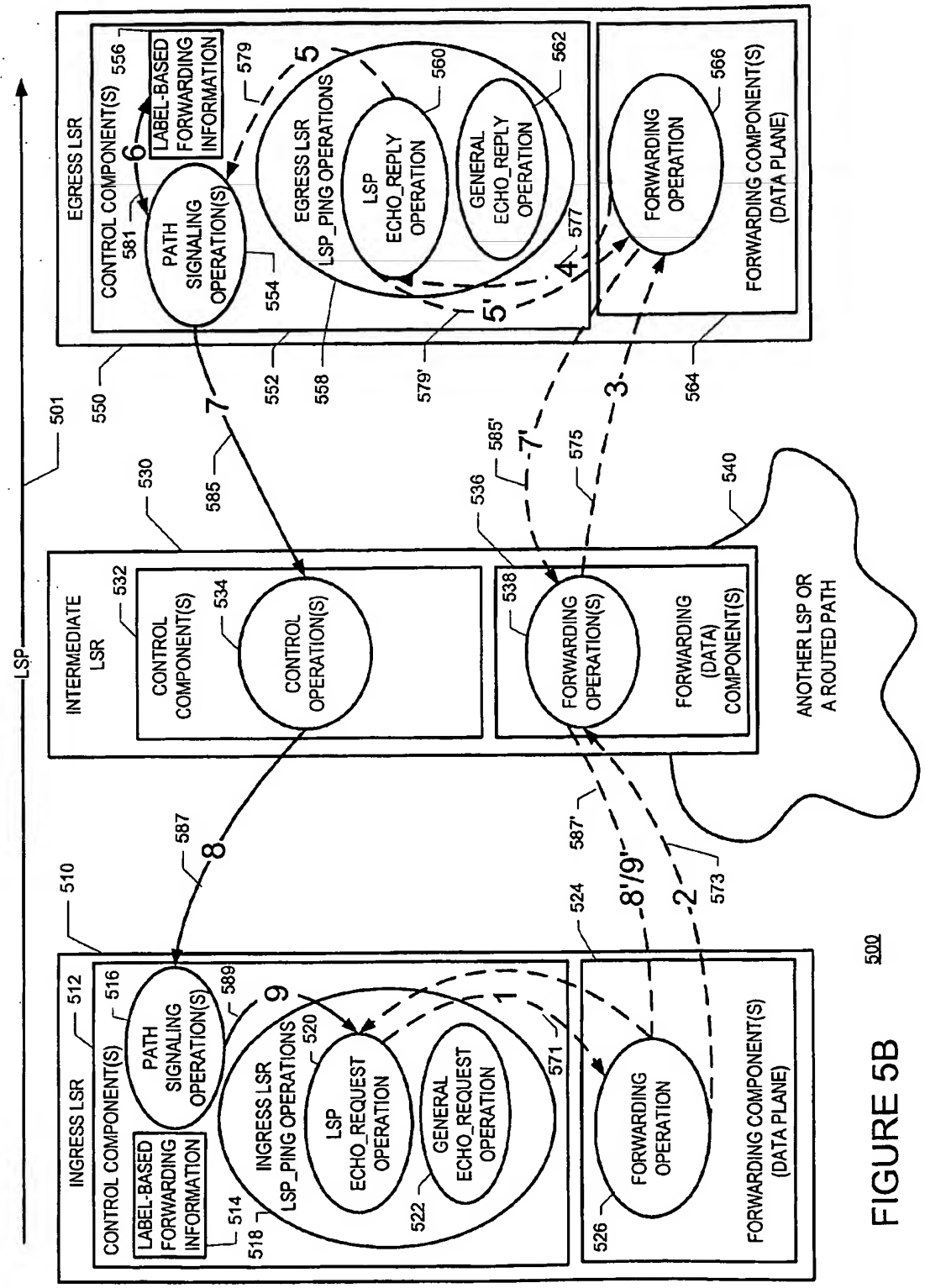
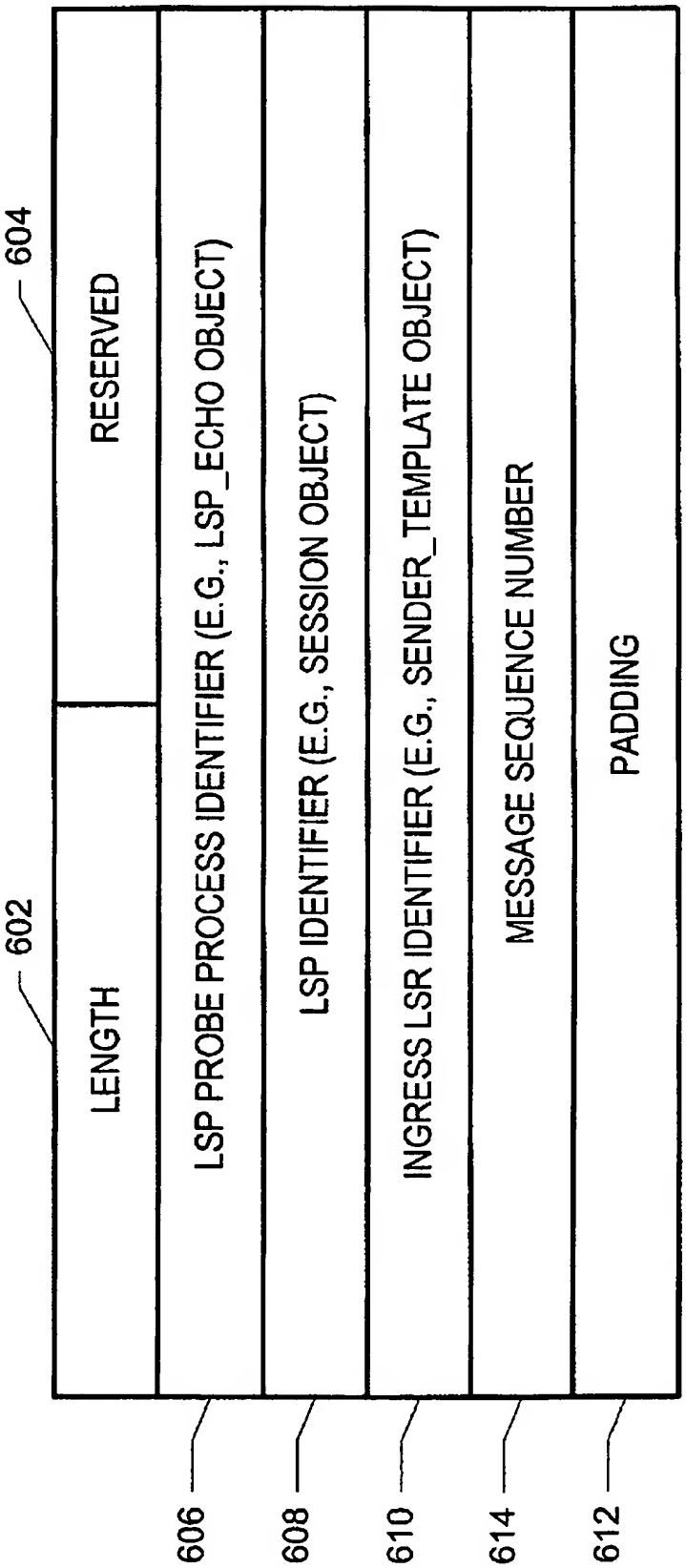

EGRESS LSR
CONTROL COMPONENT(S)
556
LABEL-BASED FORWARDING INFORMATION
579
PATH SIGNALING OPERATION(S)
581
554
EGRESS LSR LSP_PING OPERATIONS
560
LSP ECHO_REPLY OPERATION
562
GENERAL ECHO_REPLY OPERATION
577
FORWARDING OPERATION
566
FORWARDING COMPONENT(S) (DATA PLANE)
564
550
552
558
579'
501
LSP
INTERMEDIATE LSR
530
532
CONTROL COMPONENT(S)
534
CONTROL OPERATION(S)
536
538
FORWARDING OPERATION(S)
FORWARDING (DATA) COMPONENT(S)
540
ANOTHER LSP OR A ROUTED PATH
585
585'
575
587
587'
573
INGRESS LSR
510
512
CONTROL COMPONENT(S)
516
PATH SIGNALING OPERATION(S)
589
LABEL-BASED FORWARDING INFORMATION
514
INGRESS LSR LSP_PING OPERATIONS
518
520
LSP ECHO_REQUEST OPERATION
522
GENERAL ECHO_REQUEST OPERATION
571
524
FORWARDING OPERATION
526
FORWARDING COMPONENT(S) (DATA PLANE)
1
2
3
4
5
5'
6
7
7'
8
8'/9'
9
500

FIGURE 5B


602
LENGTH
604
RESERVED
LSP PROBE PROCESS IDENTIFIER (E.G., LSP_ECHO OBJECT)
LSP IDENTIFIER (E.G., SESSION OBJECT)
INGRESS LSR IDENTIFIER (E.G., SENDER_TEMPLATE OBJECT)
MESSAGE SEQUENCE NUMBER
PADDING
606
608
610
614
612

600

FIGURE 6

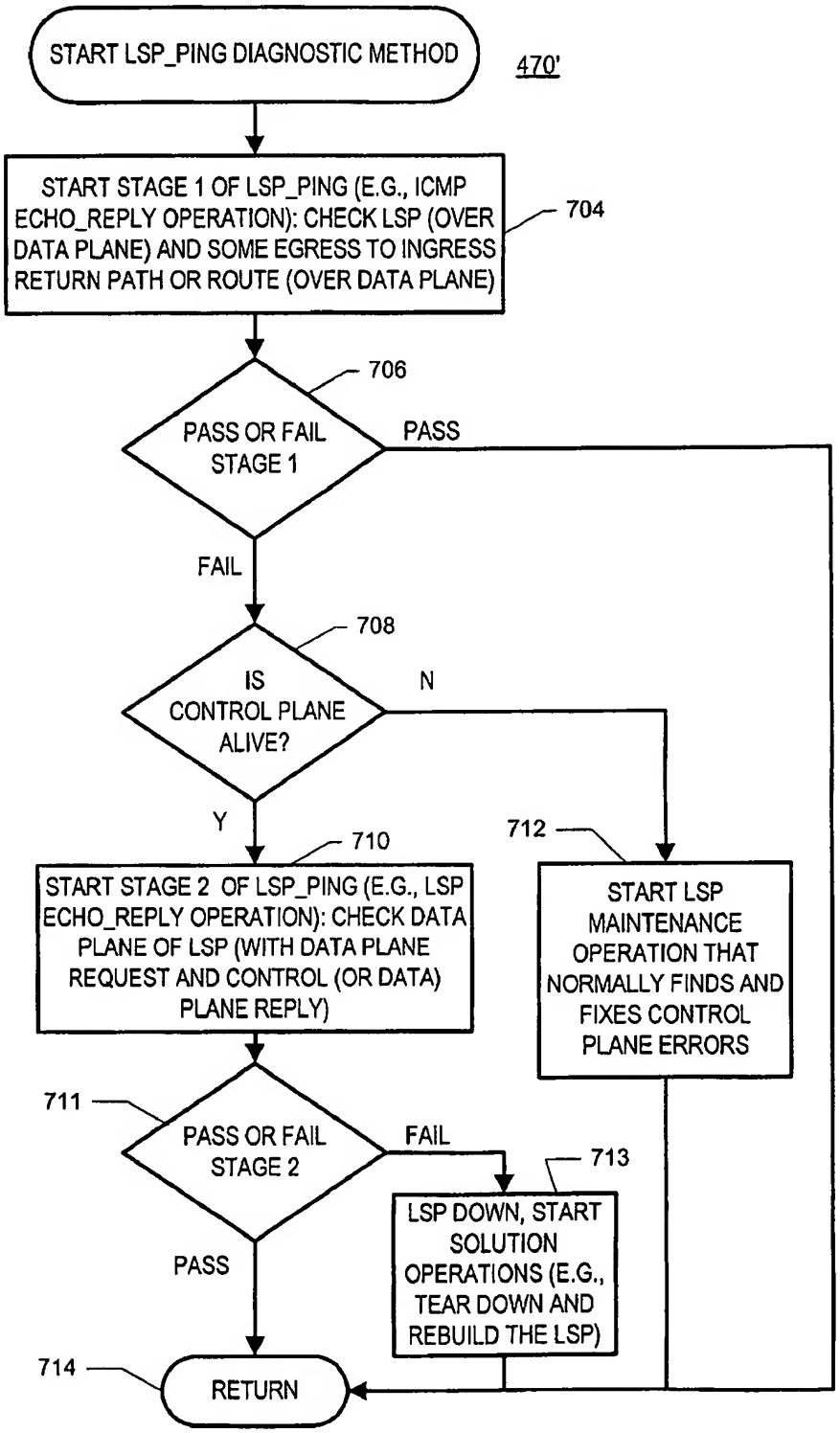

START LSP_PING DIAGNOSTIC METHOD
470'
START STAGE 1 OF LSP_PING (E.G., ICMP ECHO_REPLY OPERATION): CHECK LSP (OVER DATA PLANE) AND SOME EGRESS TO INGRESS RETURN PATH OR ROUTE (OVER DATA PLANE)
704
706
PASS OR FAIL STAGE 1
PASS
FAIL
708
IS CONTROL PLANE ALIVE?
N
Y
710
START STAGE 2 OF LSP_PING (E.G., LSP ECHO_REPLY OPERATION): CHECK DATA PLANE OF LSP (WITH DATA PLANE REQUEST AND CONTROL (OR DATA) PLANE REPLY)
712
START LSP MAINTENANCE OPERATION THAT NORMALLY FINDS AND FIXES CONTROL PLANE ERRORS
711
PASS OR FAIL STAGE 2
FAIL
713
LSP DOWN, START SOLUTION OPERATIONS (E.G., TEAR DOWN AND REBUILD THE LSP)
PASS
714
RETURN

FIGURE 7

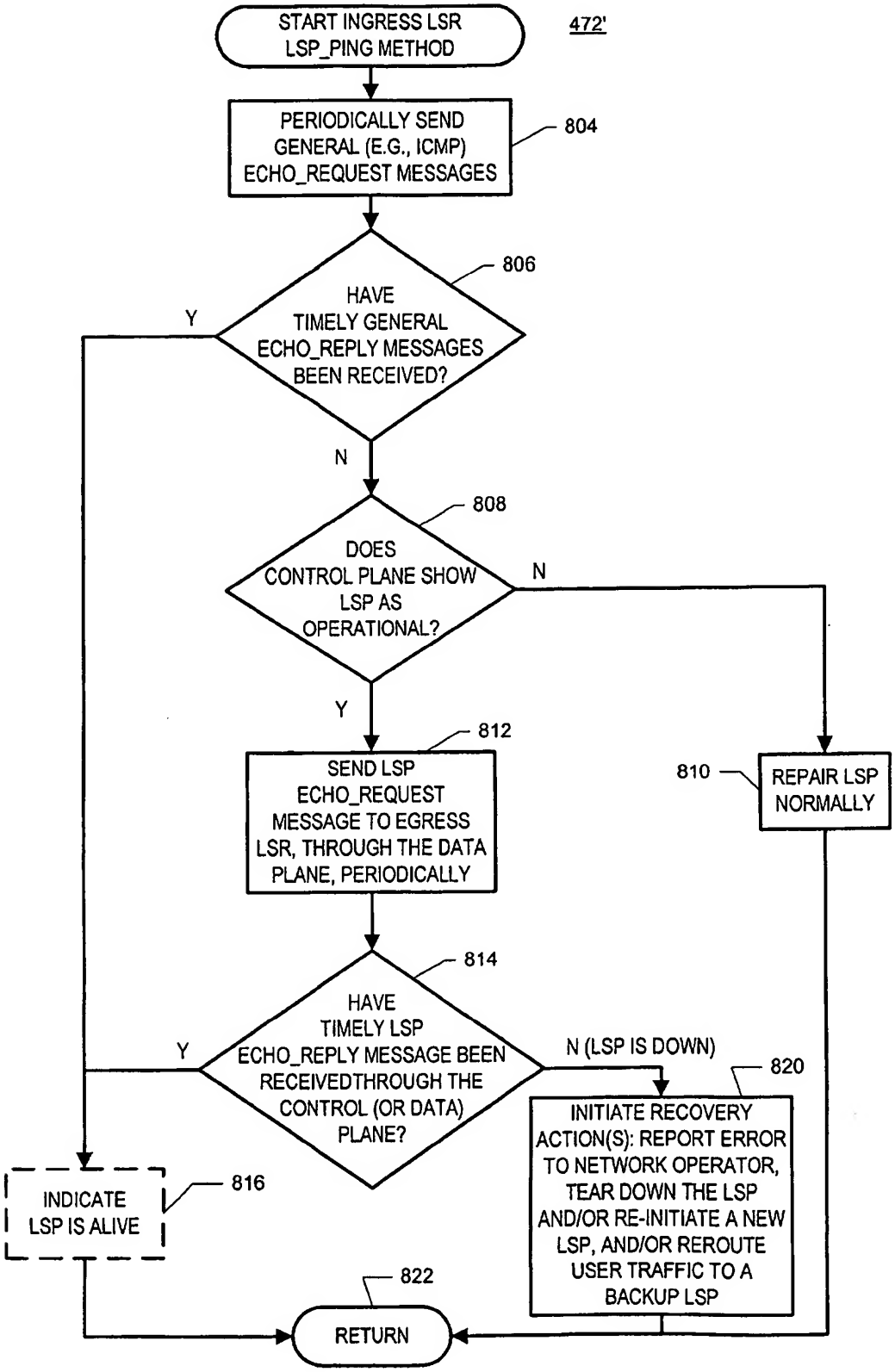

START INGRESS LSR LSP_PING METHOD
472'
PERIODICALLY SEND GENERAL (E.G., ICMP) ECHO_REQUEST MESSAGES
804
HAVE TIMELY GENERAL ECHO_REPLY MESSAGES BEEN RECEIVED?
806
Y
N
DOES CONTROL PLANE SHOW LSP AS OPERATIONAL?
808
N
Y
SEND LSP ECHO_REQUEST MESSAGE TO EGRESS LSR, THROUGH THE DATA PLANE, PERIODICALLY
812
REPAIR LSP NORMALLY
810
HAVE TIMELY LSP ECHO_REPLY MESSAGE BEEN RECEIVEDTHROUGH THE CONTROL (OR DATA) PLANE?
814
Y
N (LSP IS DOWN)
INITIATE RECOVERY ACTION(S): REPORT ERROR TO NETWORK OPERATOR, TEAR DOWN THE LSP AND/OR RE-INITIATE A NEW LSP, AND/OR REROUTE USER TRAFFIC TO A BACKUP LSP
820
INDICATE LSP IS ALIVE
816
RETURN
822

FIGURE 8

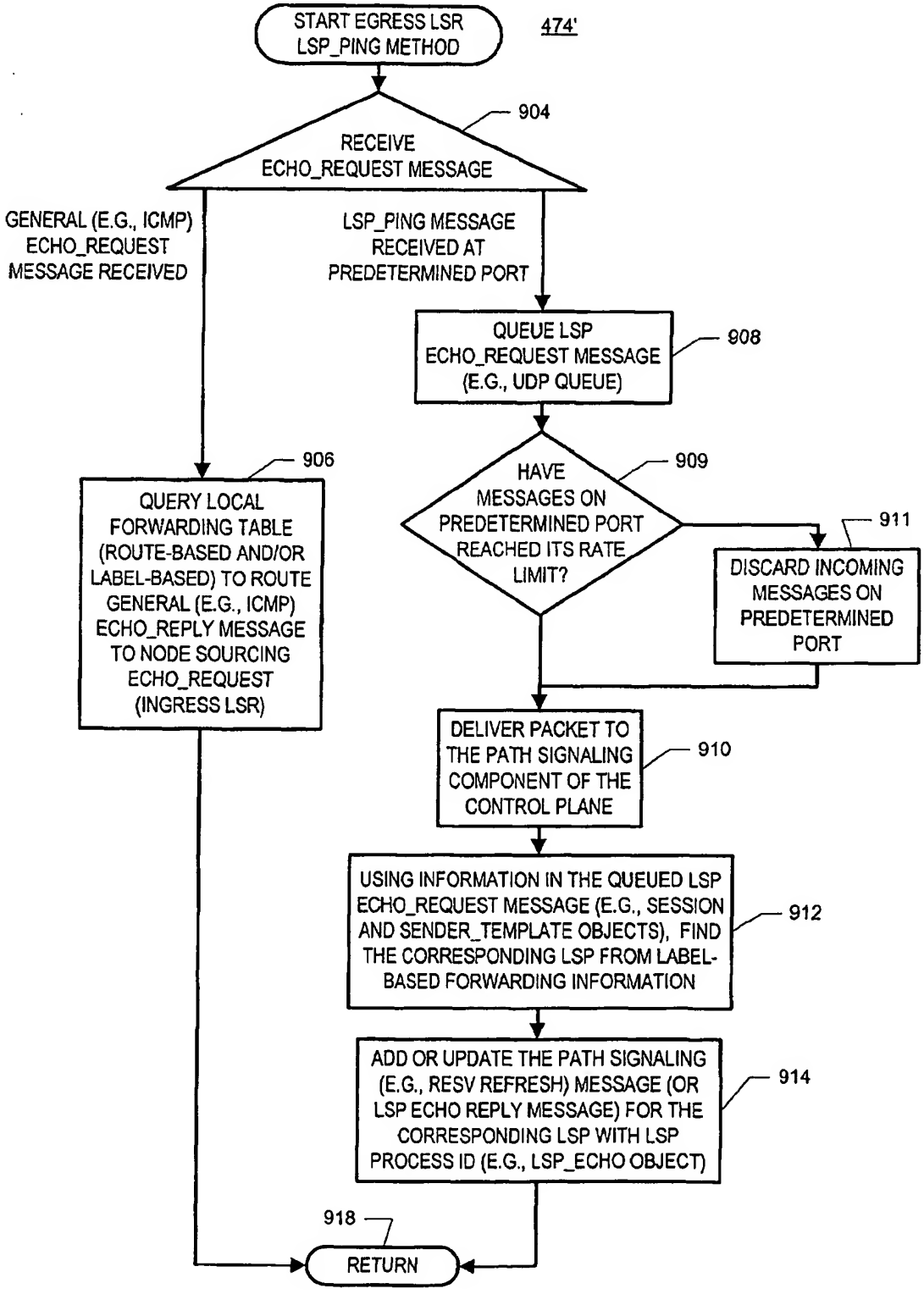

START EGRESS LSR LSP_PING METHOD
474'
RECEIVE ECHO_REQUEST MESSAGE
904
GENERAL (E.G., ICMP) ECHO_REQUEST MESSAGE RECEIVED
LSP_PING MESSAGE RECEIVED AT PREDETERMINED PORT
QUEUE LSP ECHO_REQUEST MESSAGE (E.G., UDP QUEUE)
908
HAVE MESSAGES ON PREDETERMINED PORT REACHED ITS RATE LIMIT?
909
911
DISCARD INCOMING MESSAGES ON PREDETERMINED PORT
906
QUERY LOCAL FORWARDING TABLE (ROUTE-BASED AND/OR LABEL-BASED) TO ROUTE GENERAL (E.G., ICMP) ECHO_REPLY MESSAGE TO NODE SOURCING ECHO_REQUEST (INGRESS LSR)
DELIVER PACKET TO THE PATH SIGNALING COMPONENT OF THE CONTROL PLANE
910
USING INFORMATION IN THE QUEUED LSP ECHO_REQUEST MESSAGE (E.G., SESSION AND SENDER_TEMPLATE OBJECTS), FIND THE CORRESPONDING LSP FROM LABEL-BASED FORWARDING INFORMATION
912
ADD OR UPDATE THE PATH SIGNALING (E.G., RESV REFRESH) MESSAGE (OR LSP ECHO REPLY MESSAGE) FOR THE CORRESPONDING LSP WITH LSP PROCESS ID (E.G., LSP_ECHO OBJECT)
914
918
RETURN

FIGURE 9

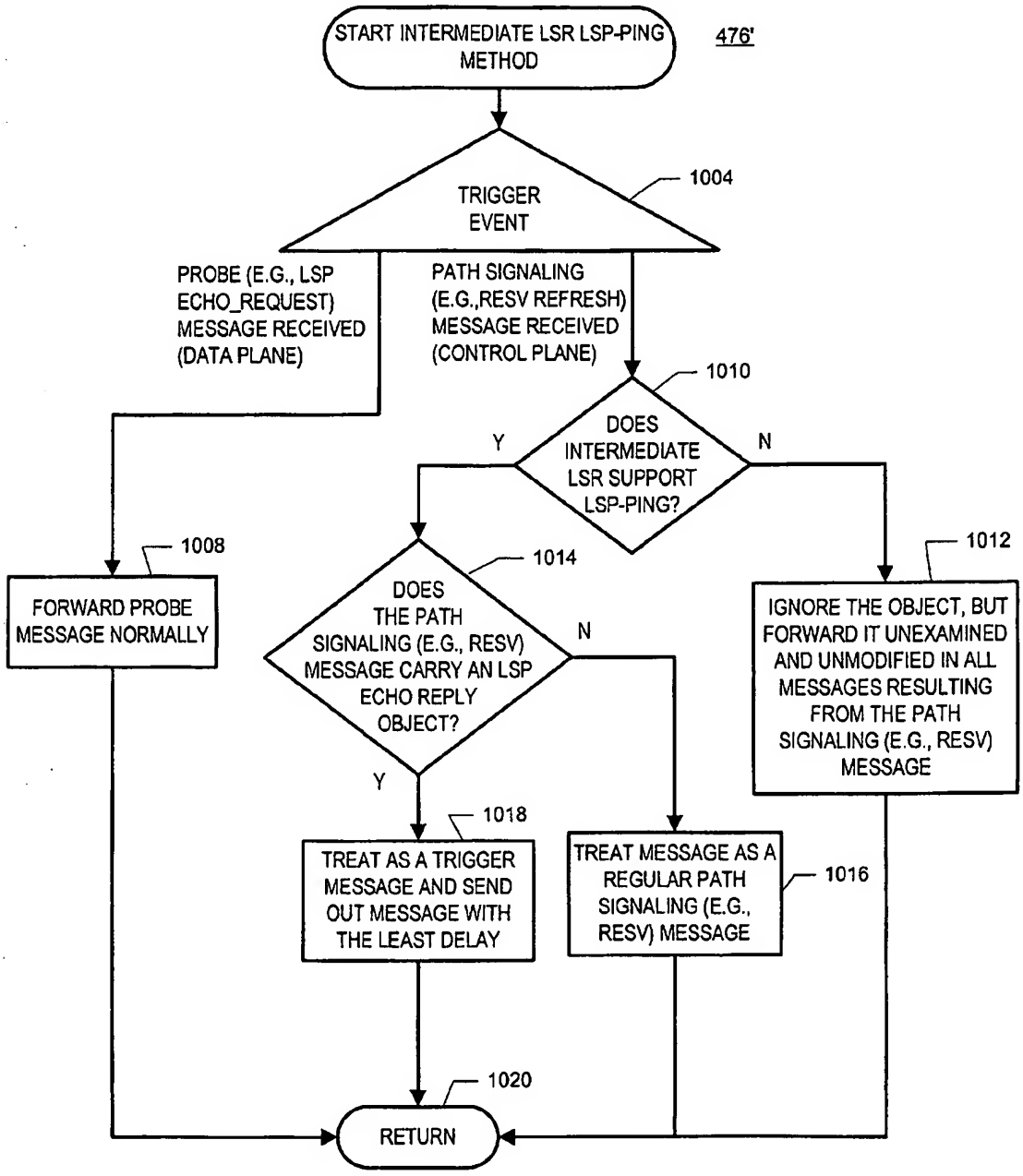
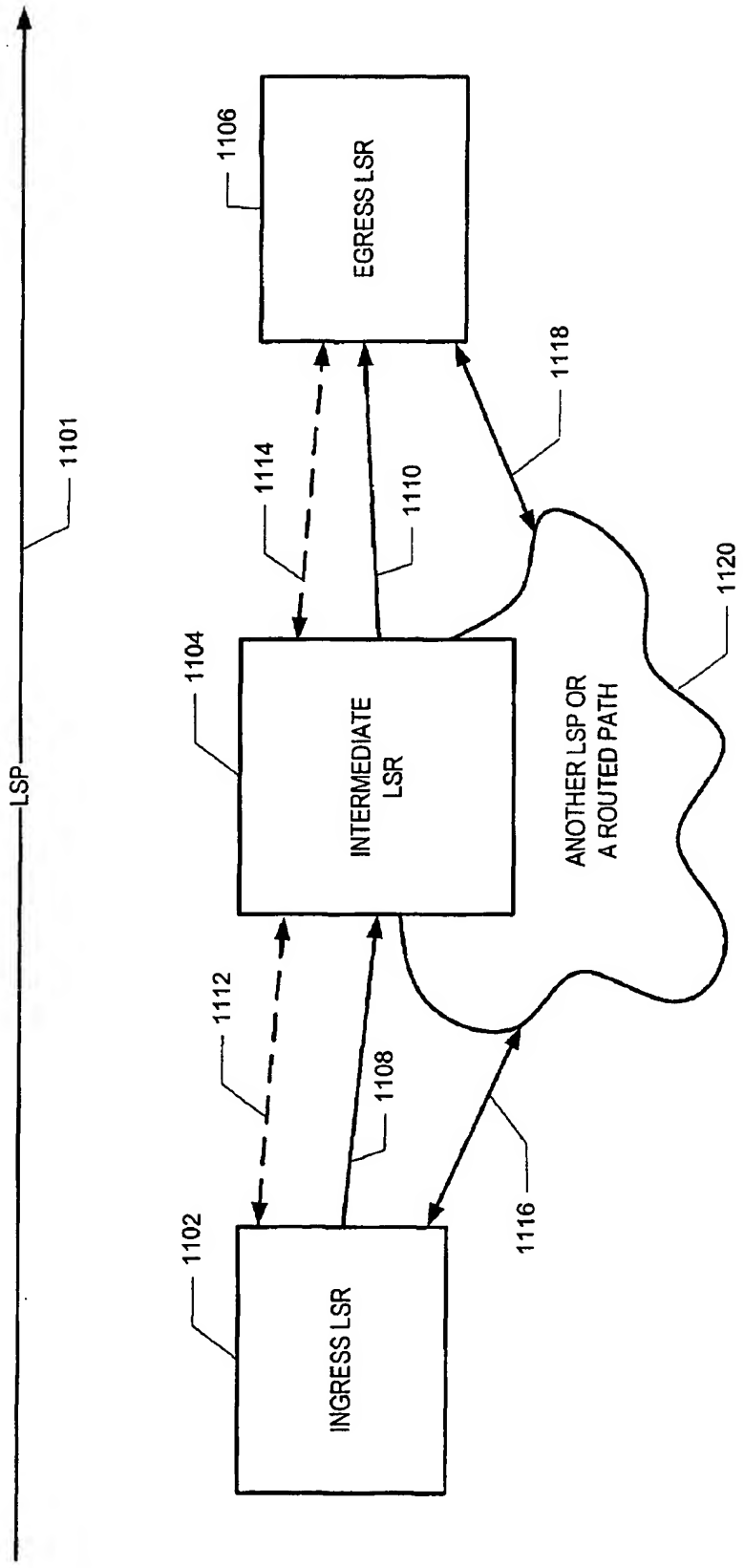

START INTERMEDIATE LSR LSP-PING METHOD
476'
TRIGGER EVENT
1004
PROBE (E.G., LSP ECHO_REQUEST) MESSAGE RECEIVED (DATA PLANE)
PATH SIGNALING (E.G.,RESV REFRESH) MESSAGE RECEIVED (CONTROL PLANE)
1010
DOES INTERMEDIATE LSR SUPPORT LSP-PING?
Y
N
1008
FORWARD PROBE MESSAGE NORMALLY
1014
DOES THE PATH SIGNALING (E.G., RESV) MESSAGE CARRY AN LSP ECHO REPLY OBJECT?
N
Y
1012
IGNORE THE OBJECT, BUT FORWARD IT UNEXAMINED AND UNMODIFIED IN ALL MESSAGES RESULTING FROM THE PATH SIGNALING (E.G., RESV) MESSAGE
1018
TREAT AS A TRIGGER MESSAGE AND SEND OUT MESSAGE WITH THE LEAST DELAY
TREAT MESSAGE AS A REGULAR PATH SIGNALING (E.G., RESV) MESSAGE
1016
1020
RETURN

FIGURE 10



1100

FIGURE 11

DETECTING DATA PLANE LIVELINES IN CONNECTIONS SUCH AS LABEL-SWITCHED PATHS

§ 0. RELATED APPLICATIONS

Benefit is claimed, under 35 U.S.C. § 119(e)(1), to the filing date of provisional patent application Ser. No. 60/301,050, entitled "DETECTING DATA PLANE LIVELINESS IN RSVP-TE", filed on Jun. 25, 2001 and listing Ping Pan and Nischal Sheth as the inventors, for any inventions disclosed in the manner provided by 35 U.S.C. § 112, ¶ 1. This provisional application is expressly incorporated herein by reference. However, the invention is not intended to be limited by any statements in that provisional application. Rather, that provisional application should be considered to describe exemplary embodiments of the invention.

§ 1. BACKGROUND OF THE INVENTION

§ 1.1 Field of the Invention

The present invention concerns detecting and diagnosing errors in connections, such as label-switched paths, that prevent user traffic from being delivered.

§ 1.2 Description of Related Art

The description of art in this section is not, and should not be interpreted to be, an admission that such art is prior art to the present invention. Although one skilled in the art will be familiar with networking, circuit switching, packet switching, label switched paths, and protocols such as RSVP and LDP, each is briefly introduced below for the convenience of the less skilled reader. More specifically, circuit switched and packet switched networks are introduced in § 1.2.1. The need for label switched paths, their operation, and their establishment are introduced in §§ 1.2.2-1.2.4 below. Finally, "failures" in a label switched path, as well as typical failure responses, are introduced in § 1.2.5 below.

§ 1.2.1 Circuit Switched Networks and Packet Switched Networks

Circuit switched networks establish a connection between hosts (parties to a communication) for the duration of their communication ("call"). The public switched telephone network ("PSTN") is an example of a circuit switched network, where parties to a call are provided with a connection for the duration of the call. Unfortunately, many communications applications, circuit switched networks use network resources inefficiently. Consider for example, the communications of short, infrequent "bursts" of data between hosts. Providing a connection for the duration of a call between such hosts simply wastes communications resources when no data is being transferred. The desire to avoid such inefficiencies has lead to the development of packet switched networks.

Packet switched networks forward addressed data (referred to as "packets" in the specification below without loss of generality), typically on a best efforts basis, from a source to a destination. Many large packet switched networks are made up of interconnected nodes (referred to as "routers" in the specification below without loss of generality). The routers may be geographically distributed throughout a region and connected by links (e.g., optical fiber, copper cable, wireless transmission channels, etc.). In such a network, each router typically interfaces with (e.g., terminates) multiple links.

Packets traverse the network by being forwarded from router to router until they reach their destinations (as typically specified by so-called layer-3 addresses in the packet headers). Unlike switches, which establish a connection for the duration of a "call" or "session" to send data received on a given input port out on a given output port, routers determine the destination addresses of received packets and, based on these destination addresses, determine, in each case, the appropriate link on which to send them. Routers may use protocols to discover the topology of the network, and algorithms to determined the most efficient ways to forward packets towards a particular destination address(es). Since the network topology can change, packets destined for the same address may be routed differently. Such packets can even arrive out of sequence.

§ 1.2.2 The Need for Label Switched Paths

In some cases, it may be considered desirable to establish a fixed path through at least a part of a packet switched network for at least some packets. More specifically, merely using known routing protocols (e.g., shortest path algorithms) to determine paths is becoming unacceptable in light of the ever-increasing volume of Internet traffic and the mission-critical nature of some Internet applications. Such known routing protocols can actually contribute to network congestion if they do not account for bandwidth availability and traffic characteristics when constructing routing (and forwarding) tables.

Traffic engineering permits network administrators to map traffic flows onto an existing physical topology. In this way, network administrators can move traffic flows away from congested shortest paths to a less congested path. Alternatively, paths can be determined autonomously, even on demand. Label-switching can be used to establish a fixed path from a head-end node (e.g., an ingress router) to a tail-end node (e.g., an egress router). The fixed path may be determined using known protocols such as RSVP or LDP. Once a path is determined, each router in the path may be configured (manually, or via some signaling mechanism) to forward packets to a peer (e.g., a "downstream" or "upstream" neighbor) router in the path. Routers in the path determine that a given set of packets (e.g., a flow) are to be sent over the fixed path (as opposed to being routed individually) based on unique labels added to the packets.

§ 1.2.3 Operations of Label Switched Paths

In one exemplary embodiment, the virtual link generated is a label-switched path ("LSP"). More specifically, recognizing that the operation of forwarding a packet, based on address information, to a next hop can be thought of as two steps-partitioning the entire set of possible packets into a set of forwarding equivalence classes ("FECs"), and mapping each FEC to a next hop. As far as the forwarding decision is concerned, different packets which get mapped to the same FEC are indistinguishable. In one technique concerning label switched paths, dubbed "multiprotocol label switching" (or "MPLS"), a particular packet is assigned to a particular FEC just once, as the packet enters the label-switched domain (part of the) network. The FEC to which the packet is assigned is encoded as a label, typically a short, fixed length value. Thus, at subsequent nodes, no further header analysis need be done—all subsequent forwarding over the label-switched domain is driven by the labels. Such FECs may be generalized such that particular ports, wavelengths, time slots, channels, etc. are used instead of labels.

FIG. **1** illustrates a label-switched path **110** across a network. Notice that label-switched paths **110** are simplex—traffic flows in one direction from a head-end label-switching router (or "LSR") **120** at an ingress edge to a tail-end label-switching router **130** at an egress edge. Generally, duplex traffic requires two label-switched paths—one for each direction. However, some protocols support bi-direc-

tional label-switched paths. Notice that a label-switched path **110** is defined by the concatenation of one or more label-switched hops, allowing a packet to be forwarded from one label-switching router (LSR) to another across the MPLS domain **110**.

Recall that a label may be a short, fixed-length value carried in the packet's header to identify a forwarding equivalence class (or "FEC"). Recall further that a FEC is a set of packets that are forwarded over the same path through a network, sometimes even if their ultimate destinations are different. At the ingress edge of the network, each packet is assigned an initial label (e.g., based on all or a part of its layer **3** destination address). More specifically, referring to the example illustrated in FIG. **2**, an ingress label-switching router **210** interprets the destination address **220** of an unlabeled packet, performs a longest-match routing table lookup, maps the packet to an FEC, assigns a label **230** to the packet and forwards it to the next hop in the label-switched path.

In the MPLS domain, the label-switching routers (LSRs) **220** ignore the packet's network layer header and simply forward the packet using label-swapping. More specifically, when a labeled packet arrives at a label-switching router (LSR), the input port number and the label are used as lookup keys into an MPLS forwarding table. When a match is found, the forwarding component retrieves the associated outgoing label, the outgoing interface (or port), and the next hop address from the forwarding table. The incoming label is replaced with the outgoing label and the packet is directed to the outgoing interface for transmission to the next hop in the label-switched path. FIG. **2** illustrates such label-switching by label-switching routers (LSRs) **220***a* and **220***b*.

When the labeled packet arrives at the egress label-switching router, if the next hop is not a label-switching router, the egress label-switching router discards ("pops") the label and forwards the packet using conventional (e.g., longest-match IP) forwarding. FIG. **2** illustrates such label popping and IP forwarding by egress label-switching router **240**.

§ 1.2.4 Establishing Label Switched Paths

In the example illustrated with reference to FIG. **2**, each label-switching router had appropriate forwarding labels. However, these labels must be provided to the label-switching routers in some way.

There are two basic types of LSPs—static LSPs and protocol (e.g., LDP, RSVP, BGP) signaled LSPs. Although each type of LSP is known to those skilled in the art, each is introduced below for the reader's convenience.

With static LSPs, labels are manually assigned on all routers involved in the path. No signaling operations by the nodes are needed.

With LDP signaled LSPs, routers establish label-switched paths (LSPs) through a network by mapping network-layer routing information directly to data link layer switched paths. LDP operates in a hop-by-hop fashion as opposed to RSVP's end-to-end fashion. More specifically, LDP associates a set of destinations (route prefixes and router addresses) with each data link LSP. This set of destinations is called the Forwarding Equivalence Class ("FEC"). These destinations all share a common data link layer-switched path egress and a common unicast routing path. Each router chooses the label advertised by the next hop for the FEC and splices it to the label it advertises to all other routers. This forms a tree of LSPs that converge on the egress router.

With RSVP signaled LSPs, an ingress (i.e., head-end) router is configured. The head-end router uses (e.g., explicit path and/or path constraint) configuration information to determine the path. The egress (i.e., tail-end) and intermediate routers accept signaling information from the ingress (i.e., head-end) router. RSVP signaled LSPs maintain "soft-state" connections; meaning an RSVP signaled LSP is refreshed periodically or it is torn down. Therefore, the routers of the LSP set up and maintain the LSP cooperatively through the use of path signaling messages such as PATH messages and RESV messages.

PATH messages are sent from the ingress router to the egress router and follow the path of the LSP. RESV messages originate from the egress router, and is delivered hop-by-hop back towards the ingress router. As a PATH message travels the path of an LSP, it takes the IP address of the router it was transmitted from and stores it in the router to which it is sent. This "IP trail" left by the PATH message is used by RESV messages to return back through the LSP path. Any errors encountered when establishing and maintaining an LSP are reported back to the ingress (i.e., head-end) router.

An LSP may be considered to have two planes, a control plane and a data plane. In an RSVP signaled LSP, the PATH and RESV messages that are used to set up and maintain the LSP are transmitted on the control plane using IP addresses, and user traffic is transmitted on the data plane using labels.

§ 1.2.5 "Failures" in a Label Switched Path

In various situations the forwarding information **220** of the forwarding component **210** of intermediate label-switching routers (LSRs) may become corrupted, e.g., OUT label **230** is stored as "3" instead of "5". In this situation, data leaving LSR **210** will be "black-holed". That is, when LSR **220***a* receives the packet with an IN label of "3" it will either discard it or transmit the packet along an LSP other than the desired LSP.

Since the control plane uses IP addresses to route its messages, the control plane is still active and does not recognize the error. Therefore, the ingress LSR may continue to transmit data through the broken LSP.

If an ingress LSR continuously fails to deliver data through a given LSP, that LSP may be suspected of being down. It is desirable to provide a tool that would detect, within a reasonable amount of time, if a suspected LSP is actually down. In the art, there is no practical and quick solution known to detect the liveliness of the data plane of an LSP. Presently, if an LSP is suspected of being down, users perform manual memory dumps along several LSPs, examining the labeling information of the control plane and the data plane of the LSPs, to discover which LSP was broken. This procedure can be very time consuming and may not be a practical solution because of the length of time needed to locate the problem.

§ 2. Summary of the Invention

The present invention discloses apparatus, data structures and methods for testing the liveliness of a data plane of a label switched path (LSP). A two stage approach is used to minimize the processing overhead imposed on an LSR control plane. The first stage uses a general echo request operation that may be implemented using hardware. Therefore, the first stage does not heavily burden the control plane of the LSR. If a suspect LSP passes the first stage of the diagnostic operation, nothing more need be done. If, however, the suspect LSP fails the first stage, the diagnostic operation proceeds to a second stage.

The second stage of the diagnostic operation sends probing messages through the suspect LSP, but uses the control plane (or alternatively, the data plane, such as another LSP)

to deliver the acknowledging messages. If the suspect LSP fails the second stage of the diagnostic operation, the ingress LSR can infer that the LSP is down, and begin recovery actions.

The present invention can also be used to test MTU limits. In addition, the LSP_ping message of the present invention is encapsulated in a protocol that allows flow control, thereby protecting an LSR that can receive LSP_ping messages from DoS attacks.

§ 3. BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates an LSP including a head-end (or ingress) LSR, an intermediate LSR, and a tail-end (or egress) LSR.

FIG. **2** illustrates label assignment, switching and popping by label-switching routers of a label-switched path.

FIG. **3** is a block diagram of an apparatus that may be used to effect at least some aspects of the present invention.

FIG. **4** is a bubble chart of a router in which the present invention may be used.

FIG. **5**A is a message diagram illustrating an example of operations of a general echo stage of the invention.

FIG. **5**B is a message diagram illustrating an example of operations of an LSP echo stage of the invention.

FIG. **6** is a block diagram illustrating the data structure of an exemplary LSP echo_request probe.

FIG. **7** is a flow diagram of an exemplary method for diagnosing a suspect LSP.

FIG. **8** is a flow diagram of an exemplary method, which may be performed by an ingress LSR of an LSP, for diagnosing a suspect LSP.

FIG. **9** is a flow diagram of an exemplary method, which may be performed by an egress LSR of an LSP, for diagnosing a suspect LSP.

FIG. **10** is a flow diagram of an exemplary method, which may be performed by an intermediate LSR of an LSP, for diagnosing a suspect LSP.

FIG. **11** is a block diagram illustrating an exemplary communication system in which a diagnostic operation is performed.

§ 4. DETAILED DESCRIPTION

The present invention involves methods, apparatus and data structures for testing the liveliness of label switched paths. The following description is presented to enable one skilled in the art to make and use the invention, and is provided in the context of particular applications and their requirements. Various modifications to the disclosed embodiments will be apparent to those skilled in the art, and the general principles set forth below may be applied to other embodiments and applications. Thus, the present invention is not intended to be limited to the embodiments shown and the inventors regard their invention as the following disclosed methods, apparatus and data structures and any other patentable subject matter.

In the following, exemplary environments in which the present invention may operate are described in § 4.1. Then high-level functions that may be performed by the present invention are introduced in § 4.2. In § 4.3, exemplary operations that may be used to effect those high-level functions are described. Thereafter, exemplary apparatus, methods and data structures that may be used are described in § 4.4. Examples illustrating operations preformed by exemplary embodiments of the invention are then provided in § 4.5. Finally, some conclusions regarding the present invention are set forth in § 4.6.

§ 4.1 ENVIRONMENT IN WHICH THE PRESENT INVENTION MAY OPERATE

The present invention may be used in communication systems including nodes for forwarding addressed data, such as packets, and having a control component and a forwarding component. The present invention may be initiated by the control component of the ingress node of a label switched path. The node may be a router that supports label-switched paths.

FIG. **4** is a bubble-chart of an exemplary router **400** in which the present invention may be used. The router **400** may include a packet forwarding operation **410** (part of a "data plane") and a control (e.g., routing) operation **420** (part of a "control plane"). The packet forwarding operation **410** may forward received packets based on route-based forwarding information **450** and/or based on label-based forwarding information **490**, such as label-switched path information.

Regarding the control operations **420**, the operations and information depicted to the right of dashed line **499** are related to creating and maintaining virtual links, such as label-switched paths, while the operations and information depicted to the left of the dashed line **499** are related to creating routes. These operations and information needn't be performed and provided, respectively, on all routers of a network.

The route selection operations **430** include information distribution operations **434** and route determination operations **432**. The information distribution operations **434** may be used to discover network topology information, store it as routing information **440**, and distribute such information. The route determination operation **432** may use the routing information **440** to generate route-based forwarding information **450**.

The path creation and maintenance operation(s) **460** may include an information distribution operation **462**, a path selection/determination operation **464**, and path signaling operations **466**. The information distribution operation **462** may be used to obtain information about the network, store such information as path information (not shown), and distribute such information. The path determination/selection operation **464** may use the routing information **440**, the path information and/or configuration information **480** to generate label-based forwarding information **490**, such as label-switched paths for example. Path signaling operations **466** may be used to accept and disseminate signal label-based forwarding information (e.g., PATH and RESV messages).

The exemplary router **400** may also include diagnostic operations **470**. The diagnostic operations **470** may include ingress LSR LSP_ping operation(s) **472**, egress LSR LSP_ping operation(s) **474**, and intermediate LSR LSP_ping operation(s) **476**. The ingress LSR LSP_ping operation(s) **472** uses label-based forwarding information **490** to test suspect LSPs, if the router **400** is the ingress router of the suspect LSP. The egress LSR LSP_ping operation(s) **474** uses label-based forwarding information **490** and may also use path signaling operations **466** to respond to requests from ingress LSRs. The intermediate LSR LSP_ping operation **476** uses label-based forwarding information **490** to forward LSP_ping request or replies. Further interactions between these operations as it pertains to the present invention will be described below.

§ 4.2 HIGH-LEVEL FUNCTIONS THAT MAY BE PERFORMED BY THE PRESENT INVENTION

One high-level function of the present invention may be to test the liveliness of the data plane of an LSP in a reasonable amount of time, while minimizing processing burdens to the control plane of the LSP. This LSP test may be used when the failure of an LSP to deliver user traffic is not detected by the control plane. For instance when the forwarding labels in the data plane of an LSR become corrupted.

To provide a relatively quick test without adding extra burdens to the control plane of an LSP, the present invention may use a multistage diagnostic approach. Before effecting the diagnostic operation, it should be determined if both ingress and egress LSRs support the diagnostic operation, though such a check is not necessary.

The first stage of the diagnostic operation implements a general echo operation that may be, and in some embodiments is, performed at the hardware level of a router. Therefore, in routers employing a hardware-intensive forwarding component (data plane) and a software-intensive control component (control plane), the first stage of the diagnostic operation has little impact on the performance of the LSP's control plane. The ingress LSR periodically sends a general echo request through the suspect LSP towards an egress LSR. When the egress LSR receives the general echo request, it generates a general echo reply, and sends the reply to the sourcing LSR using conventional methods (e.g., it may use another LSP, or a routed path). If the ingress LSR receives the general echo reply, it can infer that the suspect LSP is alive, and nothing more is done. If, on the other hand, the ingress LSR does not receive a general echo reply in a predetermined amount of time, the ingress LSR can infer that an error occurred, either in the LSP, the reverse path, or both. To diagnose the source of the error, the diagnostic operation proceeds to a second stage.

The second stage of the diagnostic operation also transmits data through the LSP to test it, but it uses the control plane (or alternatively (e.g., another LSP of) the data plane) to send acknowledgement messages back to the sourcing LSR. The control plane has operations to detect when an error occurs in the control plane of an LSP. If an error in the control plane of an LSP occurs, the ingress LSR will know about the error through the occurrence of error messages and/or the absence of expected messages. If these error messages are not received and/or the expected messages are received, it may be inferred that the control plane of the LSP is alive. Since it is known that the control plane of the LSP is alive, it may be used as a return path when testing the liveliness of the data plane of the LSP. As stated above, alternatively (another LSP of) the data plane may be used as a return path for second stage acknowledgement messages.

The second stage of the diagnostic operation proceeds by sending a probing packet through the data plane of the suspect LSP. Once the egress LSR receives the probing packet, it returns an acknowledgement back to the sourcing ingress LSR through the control plane (or alternatively, via (another LSP of) the regular data forwarding plane). If an acknowledgment is received within a predetermined amount of time, the ingress LSR can infer that the suspect LSP is alive. If the ingress LSR does not receive an acknowledgement, then the ingress LSR may infer that the suspect LSP is down, and recovery actions are started. These recovery actions may include informing a network operator of the problem and/or rerouting user traffic to a back-up LSP.

The present invention is not limited to occasions when an LSP fails to deliver user traffic. Another high-level function that may be served by the probing packet is to test if the LSP can handle an advertised maximum transmission unit (MTU). The probing packet of the second stage of the diagnostic operation may be padded with extra bits to conform to the advertised MTU. If the ingress LSR receives an acknowledgment from the egress LSR, then the probing packet was not dropped along the LSP, and the LSP can of deliver packets the size of the advertised MTU.

In addition, to protect against Denial of Service (DoS) attacks, the probing packet (e.g., the LSP_ping message) should be encapsulated in a protocol that offers flow control mechanisms. This limits the damage that may be caused if an LSR is flooded with probing packets, which is typical in a DoS attack.

§ 4.3 EXEMPLARY OPERATIONS

FIG. **5A** is a messaging diagram illustrating the first stage in an exemplary implementation of the diagnostic operation of the present invention. FIG. **5B** is a messaging diagram illustrating the second stage in an exemplary implementation of the diagnostic operation of the present invention. The discussion of the diagnostic operation will begin with a description of communication system **500** and the steps of the first stage in § 4.3.1. Then the steps of the second stage are discussed in § 4.3.2. In § 4.3.3 using flow control to protect against DoS attacks is discussed.

§ 4.3.1 Communications System Description and Stage One Operations

In communications system **500**, an LSP **501** is defined to pass through an ingress LSR **510**, one (or more) intermediate LSR **530**, and an egress LSR **550**. The communications system **500** may include other network components, represented as a cloud **540**, that may or may not be part of the LSP **501**.

LSR **510** is an exemplary implementation of exemplary router **400** and defines the ingress of LSP **501**. Ingress LSR **510** includes control component(s) **512** and forwarding component(s) **524**.

Control component(s) **512** includes label-based forwarding information **514**, a path signaling operation(s) **516**, and an ingress LSR LSP_ping operation(s) **518**. The label-based forwarding information **514** and the path signaling operation(s) **516** both serve the same or similar functions as the label-based forwarding information **490** and the path signaling operation(s) **466** of exemplary router **400**.

Ingress LSR LSP_ping operation **518** includes an LSP echo_request operation **520** and a general echo_request operation **522**. General echo_request operation **522** implements part of the first stage of the diagnostic operation of the present invention. LSP echo_request operation **520** implements part of the second stage of the diagnostic operation of the present invention. These operations will be discussed in more detail below.

The forwarding component **524** includes a forwarding operation(s) **526**, which performs the same or similar functions as forwarding operations **410** of exemplary router **400**.

The intermediate LSR **530** includes a control component(s) **532**, and a forwarding component(s) **536**. The control component(s) **532** includes a control operation(s) **534**, which forwards path signaling messages through LSP **501**. The forwarding component(s) **536** includes a forwarding operation(s) **538**, which forwards user traffic on the data

plane of LSP **501**. The exemplary communications network **500** shows one intermediate LSP **530**. However, the present invention may be used with LSPs that include a plurality of intermediate LSRs, or alternatively, no intermediate LSRs.

LSR **550** is an exemplary implementation of exemplary router **400** and defines the egress of LSP **501**. Egress LSR **550** includes control component(s) **552** and forwarding component(s) **564**.

Control component(s) **552** includes label-based forwarding information **556**, a path signaling operation(s) **554**, and egress LSR LSP_ping operations **558**. The label-based forwarding information **556** and the path signaling operation **554** both serve the same or similar functions as the label-based forwarding information **490** and the path signaling operation **466** of exemplary router **400**.

The egress LSR LSP_ping operations **558** includes an LSP echo_reply operation **560** and a general echo_reply operation **562**. The general echo_reply operation **560** implements part of the first stage of the diagnostic operation of the present invention. The LSP echo_reply operation **562** implements part of the second stage of the diagnostic operation of the present invention. These operations will be discussed in more detail below.

The forwarding component(s) **564** includes a forwarding operation(s) **566**, which performs the same or similar functions as the forwarding operations **410** of exemplary router **400**.

The control component(s) **512, 532, 552** of the three LSRs **510, 530, 550** may be referred to as defining a control plane of LSP **501**. The forwarding component **524, 536, 564** of the three LSRs **510, 530, 550** may be referred to as defining a data plane of LSP **501**.

The first stage of an exemplary diagnosis operation **470** will now be discussed, with reference to FIG. **5**A. Dashed arrows represent information traveling downstream towards the egress LSR **550**, and solid arrows represent information traveling upstream towards the ingress LSR **510**.

Before initiating the diagnostic operation, it is established that both the ingress and the egress LSRs support the diagnostic operation.

The first stage of diagnosis operation **470** begins at ingress LSR **510**. The ingress LSR LSP_ping operations **518** initiates the general echo_request operation **522**. Dashed arrow **570** represents a first communication ("1") transmitted by the general echo_request operation **522**. A general echo request message, which includes a test message and an identifier, is generated and passed to the forwarding operation **526** for transmission through LSP **501**. The identifier is used by the ingress LSR to distinguish between multiple LSPs, and may be used as the test message.

The general echo request may, and in some embodiments does, use Internet Control Message Protocol (ICMP). In such embodiments, the general echo request message includes a source IP address, a destination IP address, a test message, and message-type field. When the destination node receives the echo request, it can simply switch the source and destination IP addresses, change the message-type field to label the packet as an echo reply message, and transmit the packet upstream to the sourcing LSR. This uses very little processing resources and in some embodiments, this echo reply operation is built into the hardware of an LSR.

Next, dashed arrow **572** ("2") represents the general echo request message being transmitted to the intermediate LSR **530**. The forwarding operation(s) **538** of the intermediate LSR **530** may then forward the general echo request message along LSP **501** in a conventional manner.

Dashed arrow **574** ("3") represents the general echo request message arriving at egress LSR **550**. Dashed arrow **576** ("4") represents the forwarding operation **566** sending the general echo request message to the general echo_reply operation **562**.

The general echo_reply operation **562** may process the general echo request message (e.g., ICMP echo request) as general echo requests are conventionally processed. In an ICMP embodiment, an echo reply is generated by switching the source and destination IP addresses and changing the message-type field of the packet to identify the packet as an ICMP echo reply message.

Solid arrow **578** ("5") represents the generated general echo reply message being sent to the forwarding operation **566** for transmission back to the node that sourced the echo request message. Solid arrow **580** ("6") represents the forwarding operation **566** transmitting the general echo reply message through another LSP, or via a routed path through network nodes/components **540**. Solid arrow **582** ("7") represents the general echo reply message arriving at the ingress LSR **510**. Solid arrow **584** ("8") represents the forwarding operation **526** passing the general echo reply message to the general echo_request operation **522**.

The test message and identifier of the general echo reply message are inspected, and if they match what was sent in the general echo request message, it may be inferred that LSP **501** is alive. If the LSP **501** is considered to be alive, the second stage of the diagnostic operation does not have to be performed.

During an execution of the first stage of the diagnostic operation, general echo request operation **522** may periodically transmit general echo request messages. If ingress LSR **510** does not receive general echo reply messages from egress LSR **550** for a predetermined (e.g., long) period of time, there may be a failure in the LSP, in the return path, or both. The diagnostic operation then proceeds to its second stage to determine if the failure occurred in LSP **501**.

§ 4.3.2 Stage Two Operations

Now the second stage of an exemplary diagnosis operation **470** will be discussed, with reference to FIG. **5**B. Dashed arrows represent information traveling downstream towards the egress LSR **550**, and solid arrows represent information traveling upstream towards the ingress LSR **510**. At this point of the diagnostic operation, the first stage of the diagnostic operation has indicated a failure, but it is not known with certainty whether the failure is due to a failure of the LSP **501**.

If there were failures in the control plane of an LSP, then the protocol that refreshes the LSP, e.g., RSVP, would detect the failure. Therefore, in this example, it may be assumed that the control plane of LSP **501** is still operational, though this cannot be said for the data (forwarding) plane of the LSP **501**. Although the foregoing assumes that the LSP is a soft-state LSP, the present invention can be used with non-soft-state LSPs.

Since the control plane is still alive, the second stage of the diagnosis operation uses this fact to test LSP **501**. A test message (e.g., an LSP_ping message) is sent through the data plane of LSP **501** and the control plane of LSP **501** (or alternatively, the data plane (e.g., of another LSP)) is used to return an acknowledging message. Since it is known that the control plane is alive, if no reply is returned from egress LSR **550**, it may be assumed that the LSP **501** is down.

FIG. **5**B is a messaging diagram that illustrates exemplary second stage operations. Dashed arrow **571** ("1") represents

the LSP echo_request operation **520** sending a generated LSP_ping message to forwarding operation **526** for transmission. An LSP_ping message basically includes an LSP identifier, an ingress LSR (or source) identifier, and an LSP probe process identifier (which may be included in an LSP_echo object). A message sequence number may also be provided. The fields of an exemplary, more complex, LSP_ping message **600** will be discussed later. The LSP identifier and the ingress LSR identifier are used by an egress LSR to locate the LSP in its label-based forwarding information so that the egress LSR can use its control component to send a reply back to the ingress LSR that sourced the LSR_ping message. The LSP probe process identifier (e.g., LSP_echo object) carried within the LSP_ping message is returned to the sourcing LSR by the egress LSR, and is used to verify that the LSP is alive. In addition, since it is possible for the ingress LSR to check multiple LSPs, each check possibly including multiple LSP_ping messages, the LSP probe process identifier (e.g., LSP_echo object) may also be used by the ingress LSR to distinguish between LSPs being checked, and LSP_ping messages

Dashed arrow **573** ("2") represents the LSP_ping message sent by the forwarding operation(s) **526** through LSP **501** in a conventional manner. Dashed arrow **575** ("3") represents the LSP_ping message sent by the forwarding operation **538** of intermediate LSR **530** to egress LSR **550** in a conventional manner. Dashed arrow **577** ("4") represents the LSP_ping message being sent to the LSP echo_reply operation **560**.

The LSP echo_reply operation **560** uses the LSP identifier and the ingress LSR identifier to locate LSP **501** in its label-based forwarding information **556**. In one embodiment, the LSP echo_reply operation **560** then checks to see if the path signaling, e.g., RESV, refresh message for LSP **501** already includes an LSP probe process identifier (e.g., LSP_echo object). If not, echo_reply operation **560** adds or updates the LSP probe process identifier (e.g., LSP_echo object) into the path signaling, e.g., RESV, refresh message. This is represented by dashed arrow **579** ("5").

In an embodiment of the present invention to be used with soft-state LSPs, when it is time to refresh LSP **501**, solid arrow **581** ("6") represents the path signaling operation(s) **554** accessing the label-based forwarding information **556** for the path signaling (e.g., RESV) refresh message for LSP **501**. Solid arrow **585** ("7") represents the path signaling operation **554** forwarding the path signaling (e.g., RESV) refresh message including the LSP probe process identifier (e.g., LSP_echo object) to the intermediate LSR **530**.

Solid arrow **587** ("8") represents the control operation(s) **534** of the intermediate LSR **530** forwarding the path signaling (e.g., RESV) refresh message to ingress LSR **510**. (In FIG. **5**B the reverse LSP path followed by the path signaling (e.g., RESV) refresh message is a high level depiction. Therefore, at each LSR, the control plane may send the path signaling (e.g., RESV) refresh message to its forwarding component to perform the physical action of the path signaling (e.g., RSEV) refresh message.)

Solid arrow **589** ("9") represents the path signaling operation **516** sending the LSP probe process identifier (e.g., LSP_echo object) to the LSP echo request operation **520**. The LSP echo_request operation **520** compares the LSP probe process identifier (LSP_echo object) it received to the LSP probe process identifier (LSP_echo object) it transmitted, and if they are the same, it may infer that LSP **501** is alive. Since LSR **510** may be the ingress LSR to a plurality of LSPs, and LSR **510** may be diagnosing more than one LSP, a unique LSP probe process identifier (e.g., LSP_echo object) may be assigned to each suspect LSP so that LSR **510** may distinguish them. Further, if a given LSP echo request process generates multiple probing packets, sequence numbers may be used to distinguish such probing packets.

During an execution of the second stage of a diagnostic operation, LSP echo_request operation **520** will periodically transmit LSP_ping messages. If ingress LSR **510** does not receive an LSP probe process identifier (LSP_echo objects) from egress LSR **550** for a predetermined period of time, it may assume that there is a failure in LSP **501**, and LSP **501** is declared as down. At this point, the ingress LSR **510** may try to fix and/or bypass the down LSP. This may include generating a message to network management, tearing-down and rebuilding the LSP, and/or rerouting user traffic to a backup LSP.

As indicated by alternative paths **5'**, **7'** and **8'/9'**, the LSP echo_reply or acknowledge message may be sent from the egress node **550** to the ingress node **510** via the data plane (e.g. via another LSP) in an alternative embodiment of the invention.

A further alternative method for implementing the second stage of a diagnostic operation includes sending a test message downstream, using the control plane, e.g., in a PATH message. Once the test message is received by the egress LSR **550**, it returns the test message to the sourcing ingress LSR **510** using the same upstream data path that was or would have been attempted in the first stage of the diagnostic operation. If the ingress LSR **510** receives the test message back from the egress LSR **550**, it is determined that the return path is alive. Therefore, it may be inferred that the suspect LSP **501** caused the first stage of the diagnostic operation to fail, and suspect LSP **501** can therefore be determined to be down.

§ 4.3.3 Flow Control Considerations

When the LSP_ping messages are transmitted by the LSP echo request operation **520**, each message should be encapsulated in a protocol that allows for flow control, e.g., User Datagram Protocol ("UDP"). UDP and other similar protocols use unique port numbers so that a router receiving such a message can identify the type of message it is receiving. This unique port number can be used by a rate limiter to regulate the amount of LSP_ping messages being sent to the control components of an egress LSR, thereby protecting the LSR in the event of a DoS attack.

§ 4.4 METHODS, DATA STRUCTURES, AND APPARATUS

Exemplary methods and data structures for effecting the functions summarized in § 4.2 and the operations described in § 4.3, are described in this section. More specifically, § 4.4.1.1 describes an exemplary diagnostic method, § 4.4.1.2 describes an exemplary ingress LSR LSP_ping method, § 4.4.1.3 describes an exemplary egress LSR LSP_ping method, § 4.4.1.4 describes an exemplary intermediate LSR LSP_ping method and § 4.4.2 describes an exemplary LSP_ping message. Then, exemplary apparatus that may be used to effect the functions summarized in § 4.2 are described in § 4.4.3. Finally, an example of operations described in § 4.3 is provided in § 4.5.

§ 4.4.1 Exemplary Methods

Exemplary methods for effecting the functions summarized in § 4.2 and the operations described in § 4.3, are described below.

§ 4.4.1.1 Diagnostic Method

FIG. **7** is a high-level flow diagram of an exemplary method **470'** for effecting various diagnostic operations **470**. At block **704**, an ingress LSR starts stage 1 operations of the diagnostic operation **470**. Recall that stage 1 operations were discussed in § 4.3.1. The method **470'** then proceeds to decision block, **706**. If the suspect LSP passes stage 1 of the diagnostic operation, e.g., replies to ICMP echo requests are received from the egress router, the diagnostic method **470'** is left via RETURN node **714**. In this case, the LSP is alive and functioning. Returning to decision block **706**, if the LSP fails stage 1 operations, e.g., ICMP echo requests are not answered, then the diagnostic method **470'** proceeds to decision block **708**.

At decision block **708**, the control plane of the suspect LSP is checked to see if it is still alive. If the control plane is down, the diagnostic method **470'** proceeds to block **712**. As indicated by block **712**, the control plane starts normal LSP maintenance operations that find and fix control plane errors, before the method **470'** is left via RETURN node **714**. Returning to decision block **708**, if the control plane is alive, diagnostic method **470'** proceeds to block **710**.

As indicated by block **710**, the ingress LSR starts stage 2 operations of the diagnostic operation **470**. Recall that stage 2 operations were discussed in § 4.3.2.

Diagnostic method **470'** proceeds to decision block **711**. If the suspect LSP passes stage 2 of the diagnostic operation, e.g., LSP_ping request messages are answered by the egress router, the diagnostic method **470'** is left via RETURN node **714**. The LSP is alive and functioning, and the failure of stage 1 may have been located in the return path of the general echo reply message. Returning to decision block **711**, if the LSP fails stage 2 operations, e.g., LSP_ping request messages are not answered (or such answers are not received, but other aspects of the LSP establishing and maintaining protocols seem normal), then the diagnostic method **470'** proceeds to block **713**.

As indicated by block **713**, the control plane infers that the LSP is down and starts solution operations (e.g., it may tear down and rebuild the broken LSP, and/or reroute user traffic to a backup LSP) as a consequence. Then, diagnostic method **470'** is left via RETURN node **714**.

The foregoing describes an exemplary diagnostic method **470'** as applied over an LSP. Exemplary methods that may be performed by ingress, egress and intermediate LSRs are now described in § 4.4.1.2 through § 4.4.1.4 below.

§ 4.4.1.2 Ingress LSR LSP_Ping Method

FIG. **8** is a flow diagram of an exemplary method **472'** for effecting ingress LSR LSP_ping operations **472**. As indicated by block **804**, an ingress LSR begins to periodically send general (e.g., ICMP) echo request messages downstream through an LSP to an egress LSR.

Ingress LSR LSP_ping method **472'** proceeds to decision block **806**. If timely general (e.g., ICMP) echo reply messages are received from the egress LSR, ingress LSR LSP_ping method **472'** infers that the LSP is alive, as indicated in block **816**. The method **472'** is then left via RETURN node **822**.

Referring back to decision block **806**, if timely general (e.g., ICMP) echo reply messages are not received from the egress LSR, ingress LSR LSP_ping method **472'** proceeds to decision block **808**. In decision block **808**, the control plane is checked to see if it shows the LSP to be operational. If the control plane shows that the LSP is not operational, method **472'** proceeds to block **810**, where the LSP is repaired normally by repair operations in the control plane before the method **472'** is left via RETURN node **822**.

Referring back to decision block **808**, if the control plane shows that the LSP is still alive, the ingress LSR LSP_ping method **472'** proceeds to block **812**. As indicated by block **812**, the ingress LSR periodically sends LSP echo request messages to the egress LSR, through the data plane of the LSP. Ingress LSR LSP_ping method **472'** then proceeds to decision block **814**. If timely LSP echo reply messages are received in path signaling refresh messages (e.g., RESV refresh messages) through the control plane (or alternatively, via the data plane), it may be assumed that the LSP is alive, as indicated by block **816**. The method **472'** is then left via RETURN node **822**.

Returning to decision block **814**, if timely LSP echo_reply messages are not received in path signaling refresh messages (e.g., RESV refresh messages) through the control plane (or alternatively, via the data plane), it may be assumed that the LSP is down, and the ingress LSR LSP_ping method **472'** proceeds to block **820**.

As indicated by block **820**, one or more LSP recovery operations may be initiated. Such recovery operation(s) may include reporting the error to a network operator, tearing down and rebuilding the LSP, and/or rerouting the user traffic through a back-up LSP. Then the ingress LSR LSP_ping method **472'** is left via RETURN node **822**.

§ 4.4.1.3 Egress LSR LSP_Ping Method

FIG. **9** is a flow diagram of an exemplary method **474'** for effecting egress LSR LSP_ping operations **474**. The exemplary egress LSR LSP_ping method **474'** begins when trigger event **904** occurs. Trigger events may include the receipt of a general echo request message, and the receipt of an LSP_ping message (at a predetermined port). If a general (e.g., ICMP) echo request message is received, the method **474'** proceeds to block **906**.

As indicated by block **906**, the egress LSR queries a local forwarding table for route-based and/or label-based routing information to route a general (e.g., ICMP) echo reply message to the node that sourced the echo request message. In various embodiments, the execution of this step may be preformed by the hardware of the egress LSR, thereby minimizing the impact of any processing on the control plane of the LSP. The method **474'** is then left via RETURN node **918**.

Returning to block **904**, if an LSP_ping message is received by the egress LSR, the method **474'** proceeds to block **908**. As mentioned earlier, the LSP_ping message is encapsulated in a protocol, (e.g., UDP) which assigns a predetermined port number to LSP_ping messages. As indicated by block **908**, the LSP_ping message is queued according to the protocol in which it is encapsulated. Then method **474'** proceeds to decision block **909**.

To protect against DoS attacks, a rate limiter may be applied to the predetermined port number. Therefore at decision block **909**, if the predetermined port number has reached its incoming rate limit, any LSP_ping message received beyond the rate limit is discarded as indicated by block **911**. Method **474'** then proceeds to block **910**.

Returning to decision block **909**, if the LSP_ping messages arriving on the predetermined port have not reached their limit, method **474'** proceeds to block **910** directly.

As indicated by block **910**, in one embodiment of the present invention the packet is delivered to the path signaling component of the control plane. Then, as indicated by

block **912**, information included in the LSP echo request message (e.g., session and sender template objects) is used to find the corresponding LSP from the label-based forwarding information of the egress LSR. Further as indicated by block **914**, the path signaling (e.g., RESV) refresh message belonging to the corresponding LSP is updated to include the LSP probe process identifier (e.g., LSP_echo object), which was included in the LSP echo_request message. If an identical LSP probe process identifier (e.g., LSP_echo object) already exists, the new LSP probe process identifier (e.g., LSP_echo object) is discarded. The method **474'** is then left via RETURN node **918**.

Although not shown, the egress LSR allows the path signaling operation to send the path signaling (e.g., RESV) refresh message as it would normally do. However, as indicated in block **914**, such a path signaling refresh message may include the LSP_echo object.

In an alternative embodiment, the data plane (e.g., another LSP) is used to carry LSP echo_reply messages from the egress LSR to the ingress LSR, instead of the steps shown in blocks **910**, **912**, **914**.

§ 4.4.1.4 Intermediate LSR LSP_Ping Method

FIG. **10** is a high-level flow diagram of an exemplary method **476'** for effecting intermediate LSR LSP_ping operations **476**. Exemplary intermediate LSR LSP_ping method **476'** begins when trigger event **1004** occurs. Trigger events may include the receipt of a diagnostic probe (e.g., a general echo_request message, or an LSP_ping message) and the receipt of a path signaling message (e.g., RESV message). If a diagnostic probe is received (on the data plane), the method **476'** proceeds to block **1008**.

As indicated by block **1008**, the intermediate LSR forwards the probe message normally switching the IN label with the appropriate OUT label, because it was sent through an LSP by an ingress LSR. An intermediate LSR usually does not inspect the contents of a packet traveling through an LSP. After the message is forwarded, the intermediate LSR LSP_ping method **476'** is left via RETURN node **1020**.

Returning to trigger event **1004**, if a path signaling message (e.g., a RESV refresh message) is received on the control plane, the intermediate LSR LSP_ping method **476'** proceeds to decision block **1010**.

At decision block **1010**, if it is determined that the intermediate LSR does not support LSP_ping messages, method **476'** proceeds to block **1012**. As indicated by block **1012**, the LSP_object is ignored and forwarded unexamined and unmodified. In addition, messages resulting from that path signaling message will also include the LSP_object and be forwarded unexamined and unmodified. Then, the intermediate LSR LSP_ping method **476'** is left via RETURN node **1020**.

Referring back to decision block **1010**, if it is determined that the intermediate LSR does support LSP_ping messages, the method **476'** proceeds to decision block **1014**.

At decision block **1014**, if the path signaling, (e.g., RESV) message does not carry an LSP echo reply object, the method **476'** proceeds to step **1016**, and treats the path signaling message like a regular message. Then, intermediate LSR LSP_ping method **476'** is left via RETURN node **1020**.

Referring back to decision block **1014**, if the path signaling message carries an LSP_echo object, method **476'** proceeds to block **1018**. As indicated by block **1018**, the message is treated as a trigger message and it is forwarded with the least amount of delay. Trigger messages are RSVP messages that advertise new state and/or any other information not present in earlier transmissions. Trigger messages may include messages advertising a new state, a route change that alters a reserved path, or a modification to an existing RSVP session or reservation. Since the information in a trigger message may be vital in maintaining the continuity of an LSP, these messages are forwarded with the least amount of delay. After the trigger message is forwarded, the intermediate LSR LSP_ping method **476'** is left via RETURN node **1020**.

§4.4.1.5 Alternative LSP_Ping Method

As described above, the data plane may be used to carry an LSP echo_reply to the ingress LSR. For example, if the label distribution protocol ("LDP") is used to generate and control the LSP being checked, the LSP echo_reply (or replies) may be sent back to the source (ingress) LSR via the data plane, rather than via the control plane. Such LSP echo_reply (replies) may be in the ICMP format.

§ 4.4.2 Exemplary LSP_Ping Message

FIG. **6** is an exemplary data structure **600** for effecting an LSP_ping message. The LSP_ping message may be encapsulated in a protocol that allows flow control (e.g., UDP) to protect against DoS attacks.

LSP_ping message **600** may include a first field **602** that announces the length of the message. LSP_ping message **600** may also include a second field **604** that is reserved, e.g., for future use. A third field **606** includes an LSP probe process identifier (e.g., an LSP_echo object), a fourth field includes an LSP identifier(s), a fifth field includes an ingress LSR identifier, a sixth field includes padding and an optional seventh field includes a sequence number. (Each of the fields, three to seven, are described below.)

The LSP probe process identifier (e.g., LSP_echo object) **606** includes a source identifier. This source identifier uniquely identifies the suspect LSP, thereby allowing the ingress LSR to distinguish returned responses, in the case where multiple LSPs are being tested. For example in RSVP-TE, the form of the LSP_echo object may be 11bbbbbb, and the C-Type of the message may equal 1.

The LSP identifier **608**, and the ingress LSR identifier **610** may be used by the egress LSR to identify which path signaling (e.g., RESV) refresh message it should insert the LSP probe process identifier (e.g., LSP_echo object) into (in the embodiment using the control plane for LSP echo_replies).

The padding **612** may be used as follows. The LSP_ping message can be used to test an advertised Maximum Transmission Unit (MTU) through an LSP. The advertised MTU is the largest packet that can be delivered through an LSP before fragmentation of the packet may be required. This is done by padding the LSP_ping message until it is the size of the advertised MTU, and sending the LSP_ping message through the LSP. If the LSP_echo object is returned with a subsequent path signaling (e.g., RESV) refresh message, then the sourcing LSR may infer that the LSP can transmit packets at the advertised MTU size.

A message sequence number **614** may also be provided. This sequence number can be used to distinguish messages when a given LSP probe process generates multiple LSP probe packets.

§ 4.4.3 Exemplary Apparatus

FIG. **3** is high-level block diagram of a machine **300** which may effect one or more of the operations discussed above. The machine **300** basically includes a processor(s) **310**, an input/output interface unit(s) **330**, a storage

device(s) **320**, and a system bus(es) and/or a network(s) **340** for facilitating the communication of information among the coupled elements. An input device(s) **332** and an output device(s) **334** may be coupled with the input/output interface(s) **330**. Operations of the present invention may be effected by the processor(s) **310** executing instructions. The instructions may be stored in the storage device(s) **320** and/or received via the input/output interface(s) **330**. The instructions may be functionally grouped into processing modules.

The machine **300** may be a router or a label-switching router for example. In an exemplary router, the processor(s) **310** may include a microprocessor, a network processor, and/or (e.g., custom) integrated circuit(s). In the exemplary router, the storage device(s) **320** may include ROM, RAM, SDRAM, SRAM, SSRAM, DRAM, flash drive(s), hard disk drive(s), and/or flash cards. At least some of these storage device(s) **320** may include program instructions defining an operating system, a protocol daemon, and/or other daemons. In a preferred embodiment, the methods of the present invention may be effected by a microprocessor executing stored program instructions (e.g., defining a part of the protocol daemon). At least a portion of the machine executable instructions may be stored (temporarily or more permanently) on the storage device(s) **320** and/or may be received from an external source via an input interface unit **330**. Finally, in the exemplary router, the input/output interface unit(s) **330**, input device(s) **332** and output device(s) **334** may include interfaces to terminate communications links.

Naturally, the operations of the present invention may be effected on systems other than routers. Such other systems may employ different hardware and/or software.

§ 4.5 AN EXAMPLE ILLUSTRATING OPERATIONS OF AN EXEMPLARY EMBODIMENT

The following is an example illustrating the diagnostic operation **470** in an exemplary embodiment of the present invention. FIG. **11** is a high-level diagram illustrating communications system **1100** in which diagnostic operation **470** may be performed. Communications system **1100** includes LSP **1101**, ingress LSR **1102**, intermediate LSR **1104**, egress LSR **1106**, and other network elements **1120** capable of forwarding data. Dashed arrows **1112**, **1114** represent control plane communications pertaining to LSP **1101**, and solid arrows **1108**, **1110** represent data plane communications that can occur over LSP **1101**. Solid arrows **1116**, **1118** represent other network connections the LSRs **1102**, **1104**, **1106** may have.

In this example RSVP is used to establish LSP **1101**, Internet Control Message Protocol (ICMP) is used for the general echo_request operation of the first stage of the invention, and LSP_ping messaging is used for the second stage of the present invention. Therefore LSRs **1102**, **1106** should support IP-based routing and LSP_ping messages.

This example begins when ingress router **1102** suspects that there is a problem with LSP **1101**, but the control plane messages indicate that the LSP as alive. Ingress router **1102** starts the first stage of diagnostic operation **470** and proceeds to periodically send ICMP echo_request messages downstream over the LSP's **1101** data plane **1108**, **1110**. The egress router receives the ICMP echo request_messages and sends ICMP echo_reply messages upstream to the ingress LSR that sourced the request messages (through another LSP or routed path **1120**). In this example, assume that ICMP echo-reply messages are not being received by the ingress LSR **1102** for a predetermined time interval (There is a failure in the LSP **1108**, **1110**, the reverse path **1120**, or both.). To determine if the LSP **1101** is the problem, the ingress LSR **1102** initiates stage 2 of the diagnostic operation **470**.

In stage 2 of the diagnostic operation **470**, the ingress LSR **1102** periodically sends LSP_ping messages, encapsulated in UDP, to the egress LSR **1106** through the LSP's **1101** data plane **1108**, **1110**. When the egress LSR **1106** receives the UDP packet, it recognizes it as an LSP_ping message because of it being received at a predetermined port number.

At the egress LSR **1106**, the LSP_ping message is passed to the control plane. Using information included in the LSP_ping message, the control plane locates the corresponding RESV message for the LSP in its **1106** label-based forwarding information. The RESV message may already contain an identical or similar LSP_echo object from an earlier LSP_ping message. Therefore, the control plane checks the RESV message for an identical or similar LSP_echo object, and if an identical or similar LSP_echo object is present in the RESV message, nothing happens. If not, the egress LSR **1106** adds or updates the LSP_echo object and refreshes the RESV message.

When it is time to refresh LSP **1101** the RESV message including the LSP_echo object is sent back to ingress LSR **1102** on the control plane **1114**, **1112**. At intermediate LSR **1104**, the RESV message is classified as a trigger message and is delivered upstream as soon as possible. As mentioned earlier, trigger messages are RSVP messages that advertise new state and/or any other information not present earlier transmissions.

When the ingress router receives the RESV message, it also receives the LSP_echo object. It compares the received object to the original object that was transmitted, and if they are the same, ingress LSR infers that LSP **1101** is alive.

§ 4.6 CONCLUSIONS

As can be appreciated from the foregoing disclosure, the present invention discloses apparatus, data structures and methods for testing the liveliness of a data plane of a label switched path (LSP). A two stage approach is used to minimize the processing overhead imposed on an LSR control plane. The first stage uses a general echo request operation that may be implemented using hardware. Therefore, the first stage does not heavily burden the control plane of the LSR. If a suspect LSP passes the first stage of the diagnostic operation, nothing more is done, but if the suspect LSP fails the first stage, the diagnostic operation proceeds to a second stage.

The second stage of the diagnostic operation sends probing messages through the suspect LSP, but uses the control plane (or alternatively, the data plane) to deliver the acknowledging messages. If the suspect LSP fails the second stage of the diagnostic operation, the ingress LSR can infer that the LSP is down, and begin recovery actions.

The present invention can also be used to test MTU limits. In addition, the LSP_ping message of the present invention is encapsulated in a protocol that allows flow control, thereby protecting an LSR that can receive LSP_ping messages from DoS attacks.

What is claimed is:

1. A method for testing a label switched path, the method comprising:

a) performing a first test including testing a data plane from an ingress node of the label switched path to an

egress node of the label switch path, over the label switched path, together with a data plane from the egress node of the label switched path back to the ingress node of the label switch path;

b) determining whether or not the first test was passed;

c) if it is determined that the first test was not passed, and if a control plane of the label switched path is alive, then performing a second test including testing the data plane from the ingress node of the label switched path to the egress node of the label switch path, over the label switched path, together with the control plane from the egress node of the label switched path back to the ingress node of the label switch path,

wherein if the second test was not passed, then indicating that the data plane of the label switched path is down.

2. The method of claim 1 further comprising:

d) if it is indicated that the data plane of the label switched path is down, then reporting an error in the label switched path.

3. The method of claim 1 further comprising:

d) if it is indicated that the data plane of the label switched path is down, then using an alternative label switched path from the ingress node to the egress node for forwarding data.

4. The method of claim 1 further comprising:

d) if it is indicated that the data plane of the label switched path is down, then tearing down the label switched path.

5. The method of claim 4 further comprising:

e) generating a new label switched path from the ingress node to the egress node.

6. The method of claim 1 wherein the act of performing a first test includes

i) providing, at the ingress node, a general echo request message, addressed to the egress node over the label switched path, and

ii) determining, at the ingress node, whether a general echo reply message is received within a predetermined time,

wherein if the general echo reply message is not received within the predetermined time, the first test is not passed.

7. The method of claim 1 wherein the act of performing a second test includes

i) providing, at the ingress node, a label switched path probe message;

ii) sending the label switched path probe message to the egress node over the label switched path;

iii) if the label switched path probe message is received at the egress node, then

A) generating a probe reply message at the egress node, and

B) sending the probe reply message from the egress node toward the ingress node over the control plane,

wherein if the probe reply message is not received by the ingress node within a predetermined time, the second test is not passed.

8. The method of claim 7 wherein the label switched path probe message includes

a first value identifying the ingress node,

a second value identifying the label switched path being tested, and

a third value identifying the second test.

9. The method of claim 8 wherein the probe reply message includes a value based on the third value.

10. The method of claim 8 wherein the probe reply message includes the third value.

11. The method of claim 8 wherein the probe reply message is included in a resource reservation protocol RESV message.

12. The method of claim 7 wherein the label switched path probe message is padded such that its length corresponds to the length of a maximum transmission unit associated with the label switched path.

13. The method of claim 7 wherein the label switched path probe message is encapsulated in a user datagram protocol packet.

14. The method of claim 7 wherein the label switched path probe message is provided in accordance with a rate limiting protocol.

15. A node for use in a network as a part of a label switched path, the node comprising:

a) a forwarding component for forwarding data based on forwarding information, the forwarding component defining a part of a data plane; and

b) a control component for maintaining the forwarding information, the control component defining a part of a control plane, and including diagnostic facilities adapted to

i) perform at least a part of a first test including testing the data plane from an ingress node of the label switched path to an egress node of the label switch path, over the label switched path, together with the data plane from the egress node of the label switched path back to the ingress node of the label switch path;

ii) determine whether or not the first test was passed;

iii) perform at least a part of a second test including testing the data plane from the ingress node of the label switched path to the egress node of the label switch path, over the label switched path, together with the control plane from the egress node of the label switched path back to the ingress node of the label switch path, if it is determined that the first test was not passed, and if a control plane of the label switched path is alive, and

iv) indicating that the data plane of the label switched path is down if the second test was not passed.

16. The node of claim 15 wherein the diagnostic facility is further adapted to report an error in the label switched path if it is indicated that the data plane of the label switched path is down.

17. The node of claim 15 wherein the control component includes a microprocessor executing program instructions, and

wherein the first test has a minimal impact on microprocessor.

18. A method for testing a label switched path, for use by an ingress node of the label switched path, the method comprising:

a) performing a part of a first test by

i) sending a general echo request message to an egress node of the label switched path,

ii) waiting a first predetermined time for a reply to the general echo request message, wherein if a reply to the general echo request message is received within the predetermined time, then the first test is passed, and wherein if a reply to the general echo request is not received within the predetermined time, then the first test is not passed;

b) if it is determined that the first test was not passed, and if a control plane of the label switched path is alive, then performing a part of a second test by

i) sending a label switched path probe message to the egress node of the label switched path,

ii) waiting a second predetermined time for a control plane message indicating that the egress node received the label switched path probe message, wherein if a control plane message indicating that the egress node received the label switched path probe message is received within the second predetermined time, then a data plane of the label switched path is considered alive, and wherein if a control plane message indicating that the egress node received the label switched path probe message is not received within the second predetermined time, then a data plane of the label switched path is considered down.

19. The method of claim 18 further comprising:

d) if the data plane of the label switched path is considered down, then reporting an error in the label switched path.

20. The method of claim 18 further comprising:

d) if the data plane of the label switched path is considered down, then using an alternative label switched path from the ingress node to the egress node for forwarding data.

21. The method of claim 18 further comprising:

d) if the data plane of the label switched path is considered down, then tearing down the label switched path.

22. The method of claim 21 further comprising:

e) generating a new label switched path from the ingress node to the egress node.

23. The method of claim 18 wherein the label switched path probe message includes

a first value identifying the ingress node,

a second value identifying the label switched path being tested, and

a third value identifying the second test.

24. The method of claim 23 wherein the probe reply message includes a value based on the third value.

25. The method of claim 23 wherein the probe reply message includes the third value.

26. The method of claim 18 wherein the probe reply message is included in a resource reservation protocol RESV message.

27. The method of claim 18 wherein the label switched path probe message is padded such that its length corresponds to the length of a maximum transmission unit associated with the label switched path.

28. The method of claim 18 wherein the label switched path probe message is encapsulated in a user datagram protocol packet.

29. The method of claim 19 wherein the label switched path probe message is provided in accordance with a rate limiting protocol.

30. For use in a label switched path, a node defining the ingress of the label switched path, the node comprising:

a) a forwarding component for forwarding data based on forwarding information, the forwarding component defining a part of a data plane; and

b) a control component for maintaining the forwarding information, the control component defining a part of a control plane, and including diagnostic facilities adapted to

i) perform a part of a first test by

A) sending a general echo request to an egress node of the label switched path,

B) waiting a first predetermined time for a reply to the general echo request message, wherein if a reply to the general echo request message is received within the predetermined time, then the first test is passed, and wherein if a reply to the general echo request message is not received within the predetermined time, then the first test is not passed;

ii) if it is determined that the first test was not passed, and if a control plane of the label switched path is alive, then perform a part of a second test by

A) sending a label switched path probe message to the egress node of the label switched path,

B) waiting a second predetermined time for a control plane message indicating that the egress node received the label switched path probe message, wherein if a control plane message indicating that the egress node received the label switched path probe message is received within the second predetermined time, then a data plane of the label switched path is considered alive, and wherein if a control plane message indicating that the egress node received the label switched path probe message is not received within the second predetermined time, then a data plane of the label switched path is considered down.

31. The node of claim 30 wherein the label switched path probe message includes

a first value identifying the ingress node,

a second value identifying the label switched path being tested, and

a third value identifying the second test.

32. A method for testing a label switched path, for use by an egress node of the label switched path, the method comprising:

a) performing a part of a first test by

i) accepting a general echo request message, and

ii) replying to the accepted general echo request message;

b) performing a part of a second test by

i) accepting a label switched path probe message which includes a value identifying the second test, and

ii) sending a control plane message towards the ingress node of the label switched path, wherein the control plane message includes the value identifying the second test.

33. The method of claim 32 wherein the label switched path probe message further includes a second value identifying a node that sourced the label switched path probe message, and a third value identifying the label switched path, and

wherein the control plane message is associated with the second test identified by the third value.

34. The method of claim 32 wherein the control plane message is a trigger message.

35. The method of claim 32 wherein the control plane message is a resource reservation protocol RESV message.

36. For use in a label switched path, a node defining the egress of the label switched path, the node comprising:

a) a forwarding component for forwarding data based on forwarding information, the forwarding component defining a part of a data plane; and

b) a control component for maintaining the forwarding information, the control component defining a part of a control plane, and including diagnostic facilities adapted to

i) perform a part of a first test by

A) accepting a general echo request message, and

B) replying to the accepted general echo request message;

ii) perform a part of a second test by

A) accepting a label switched path probe message which includes a value identifying the second test, and

B) sending a control plane message towards the ingress node of the label switched path, wherein the control plane message includes the value identifying the second test.

37. The node of claim 36 wherein the label switched path probe message further includes a second value identifying a node that sourced the label switched path probe message, and a third value identifying the label switched path, and

wherein the control plane message is associated with the label switched path identified by the third value.

38. The node of claim 36 wherein the control plane message is a trigger message.

39. The node of claim 36 wherein the control plane message is a resource reservation protocol RESV message.

40. A computer-readable medium storing a computer-readable message used for testing a data plane of a label switched path, the computer-readable message comprising: a) a first value from which the label switched path can be inferred; b) a second value from which an ingress node of the label switched path can be inferred; c) a third value identifying the message; and d) padding, wherein the padding is chosen such that a total length of the message corresponds to a maximum transmission unit associated with the label switched path, wherein the message, when interpreted by a processor executing stored program instructions, facilitates the testing of the data plane of the label switched path.

41. The computer-readable medium of claim 40 wherein the computer-readable message is a resource reservation protocol RESV message including a value identifying a second test with which the reply message is associated.

42. A computer-readable medium storing computer-executable instructions which, when executed by a computer, effect the method of claim 18.

43. A computer-readable medium storing computer-executable instructions which, when executed by a computer, effect the method of claim 32.

44. A method for testing a label switched path, the method comprising:

a) performing a first test including testing a data plane from an ingress node of the label switched path to an egress node of the label switch path, over the label switched path, together with a data plane from the egress node of the label switched path back to the ingress node of the label switch path;

b) determining whether or not the first test was passed;

c) if it is determined that the first test was not passed, and if a control plane of the label switched path is alive, then performing a second test including testing the data plane from the ingress node of the label switched path to the egress node of the label switch path, over the label switched path, together with a data plane from the egress node of the label switched path back to the ingress node of the label switch path,

wherein if the second test was not passed, then indicating that the data plane of the label switched path is down.

45. A node for use in a network as a part of a label switched path, the node comprising:

a) a forwarding component for forwarding data based on forwarding information, the forwarding component defining a part of a data plane; and

b) a control component for maintaining the forwarding information, the control component defining a part of a control plane, and including diagnostic facilities adapted to

i) perform at least a part of a first test including testing the data plane from an ingress node of the label switched path to an egress node of the label switch path, over the label switched path, together with the data plane from the egress node of the label switched path back to the ingress node of the label switch path;

ii) determine whether or not the first test was passed;

iii) perform at least a part of a second test including testing the data plane from the ingress node of the label switched path to the egress node of the label switch path, over the label switched path, together with a data plane from the egress node of the label switched path back to the ingress node of the label switch path, if it is determined that the first test was not passed, and if a control plane of the label switched path is alive, and

iv) indicating that the data plane of the label switched path is down if the second test was not passed.

46. A processor-readable storage device storing processor-executable instructions which, when executed by a processor, effect the method of claim 18.

47. A processor-readable storage device storing processor-executable instructions which, when executed by a processor, effect the method of claim 32.

* * * * *

US006061330A

United States Patent [19]

Johansson

[11] Patent Number: 6,061,330

[45] Date of Patent: May 9, 2000

[54] FLOW AND CONGESTION CONTROL IN PACKET SWITCHED NETWORKS

[75] Inventor: Per Gunnar Johansson, Hägersten, Sweden

[73] Assignee: Telefoanktiebolaget LM Ericsson, Stockholm, Sweden

[21] Appl. No.: 09/153,738

[22] Filed: Sep. 15, 1998

Related U.S. Application Data

[63] Continuation of application No. PCT/SE97/00439, Mar. 14, 1997.

[30] Foreign Application Priority Data

Mar. 15, 1996 [SE] Sweden 9601000

[51] Int. Cl.[7] G01R 31/08

[52] U.S. Cl. 370/229; 370/230

[58] Field of Search 370/235, 236, 370/231, 230, 232, 233, 234, 252, 253; 395/200.63, 203.65

[56] References Cited

U.S. PATENT DOCUMENTS

5,319,638	6/1994	Lin	370/60
5,457,687	10/1995	Newman	370/85.3
5,493,566	2/1996	Ljungberg et al.	370/60
5,793,747	9/1998	Kline	370/230

OTHER PUBLICATIONS

Per Johansson et al., Transient Analysis of a Closed Loop Rate Control Algorithm for ATM, Aug. 22, 1995, pp. 33–46.

Hiroyuki Ohsaki et al., Analysis of Rate–Based Congestion Control Algorithms for ATM Networks, May 1995, pp. 296–303.

Barnhart, A.W., "Example Switch Algorithm for Section 5.4 of TM Spec.", Hughes Network Systems, ATM Forum Contribution 95–0195, Feb. 1995.

Jain, Raj et al., "ERICA+ Extension to the ERICA Switch Algorithm", Ohio State University, ATM Forum Contribution 95–0195, Oct. 1995.

Primary Examiner—Michael Horabik

Assistant Examiner—Prenell Jones

Attorney, Agent, or Firm—Burns, Doane, Swecker & Mathis, L.L.P.

[57] ABSTRACT

A control system in an ATM system controls flows of data cells and flow control management cells from a number of sources to a destination over connections passing a network element. The flow control management cells are returned from the destination via the network element to their respective sources. The network element is exposed to congestion due to contention between the connections, that necessitates queuing of the connections. The flow control management cells have an explicit rate field for an explicit rate value used to limit a source maximum allowed cell rate to a specific value. An operating function uses deviations from an available rate value for lower priority cells and from a queue length reference forming a desirable queue length, to calculate a modified explicit rate value as a function of these deviations. This modified explicit rate value is introduced into the explicit rate field of the backward flow control management cells.

24 Claims, 8 Drawing Sheets

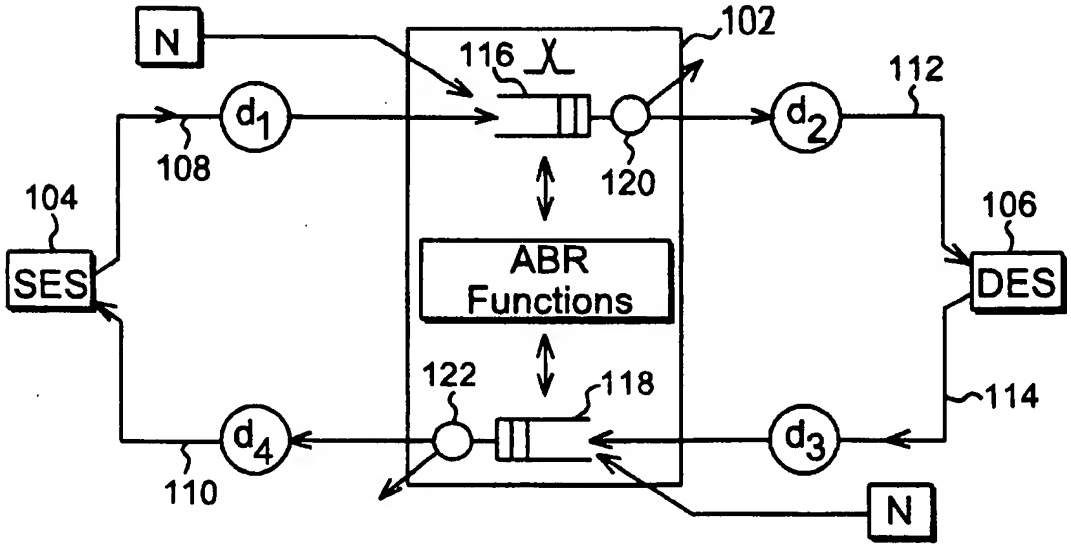


FIG. 1

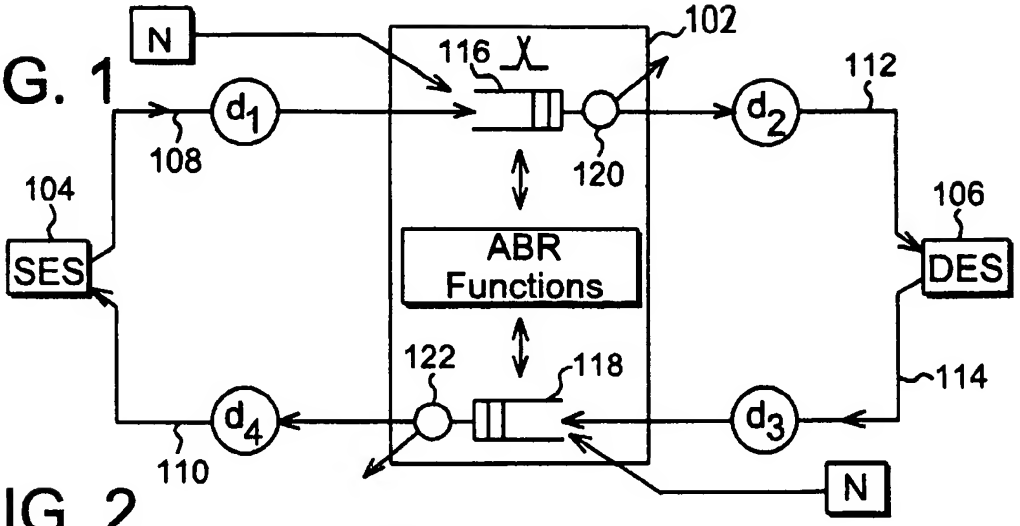


FIG. 2

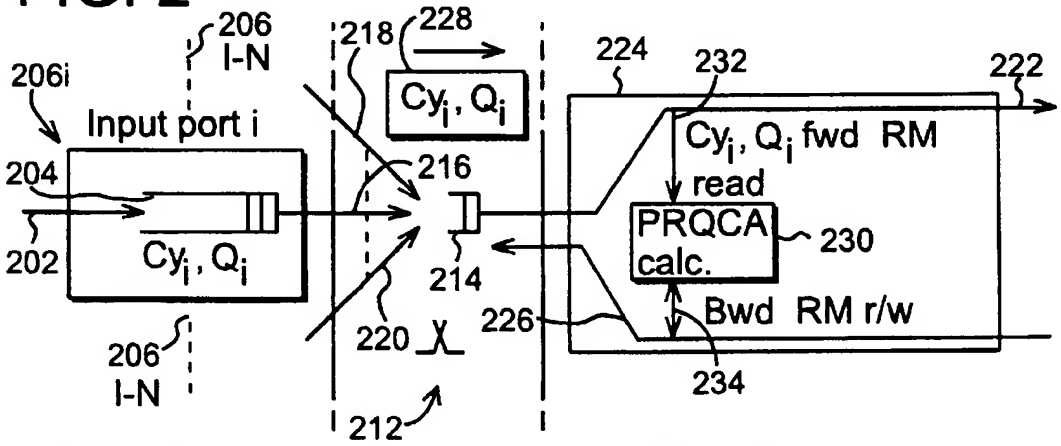


FIG. 3a

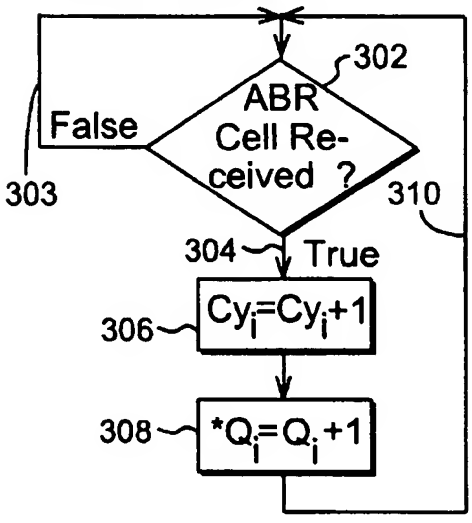
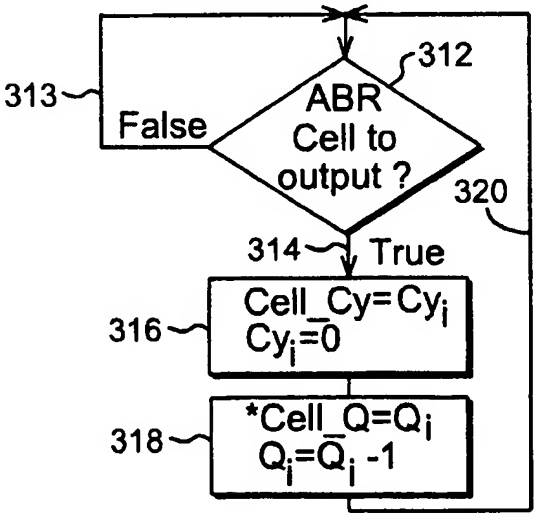


FIG. 3b



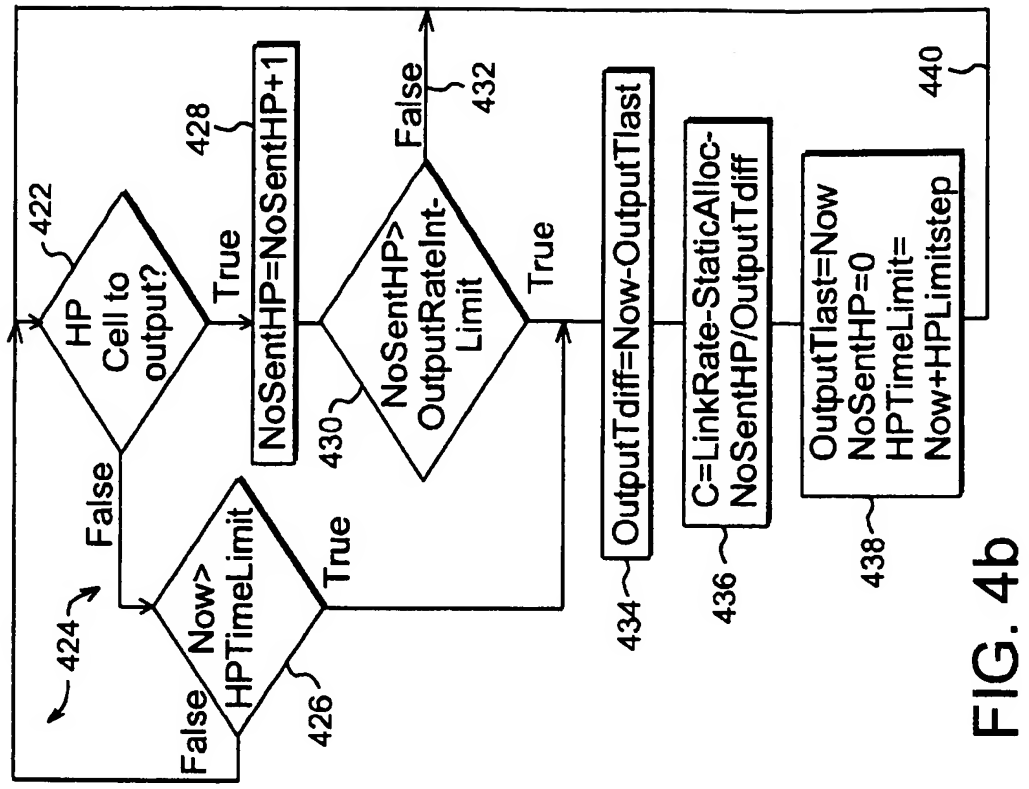

HP Cell to output?
422
True
False
NoSentHP=NoSentHP+1
428
NoSentHP> OutputRateInt-Limit
430
True
False
432
Now> HPTimeLimit
424
True
False
426
OutputTdiff=Now-OutputTlast
434
C=LinkRate-StaticAlloc-NoSentHP/OutputTdiff
436
OutputTlast=Now NoSentHP=0 HPTimeLimit= Now+HPLimitstep
438
440

FIG. 4b

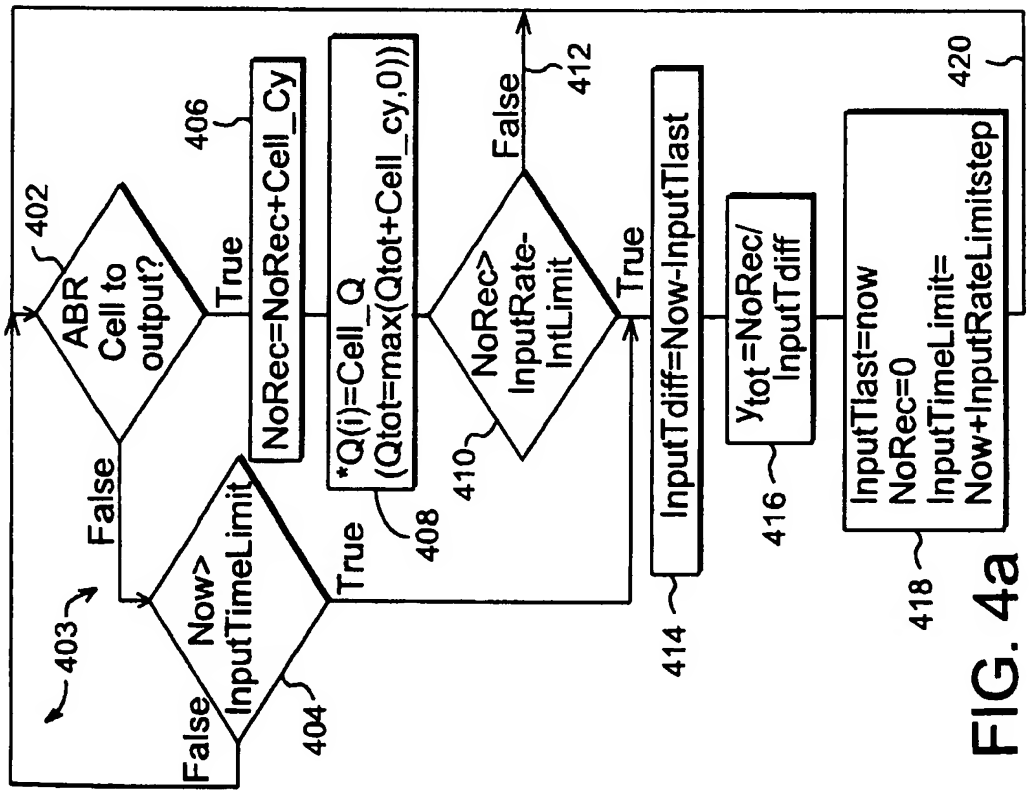

ABR Cell to output?
402
True
False
NoRec=NoRec+Cell_Cy
406
*Q(i)=Cell_Q (Qtot=max(Qtot+Cell_cy,0))
408
NoRec> InputRate-IntLimit
410
True
False
412
Now> InputTimeLimit
403
True
False
404
InputTdiff=Now-InputTlast
414
ytot=NoRec/ InputTdiff
416
InputTlast=now NoRec=0 InputTimeLimit= Now+InputRateLimitstep
418
420

FIG. 4a

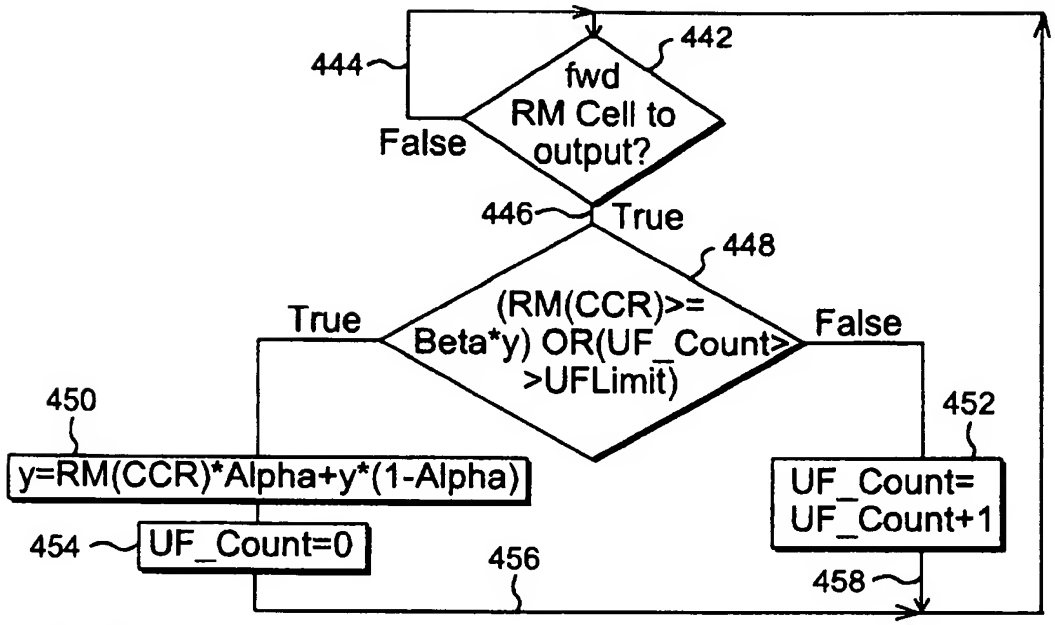

444
442
fwd
RM Cell to
output?
False
446
True
448
(RM(CCR)>=
Beta*y) OR(UF_Count>
>UFLimit)
True
False
450
452
y=RM(CCR)*Alpha+y*(1-Alpha)
UF_Count=
UF_Count+1
454
UF_Count=0
456
458

FIG. 4c

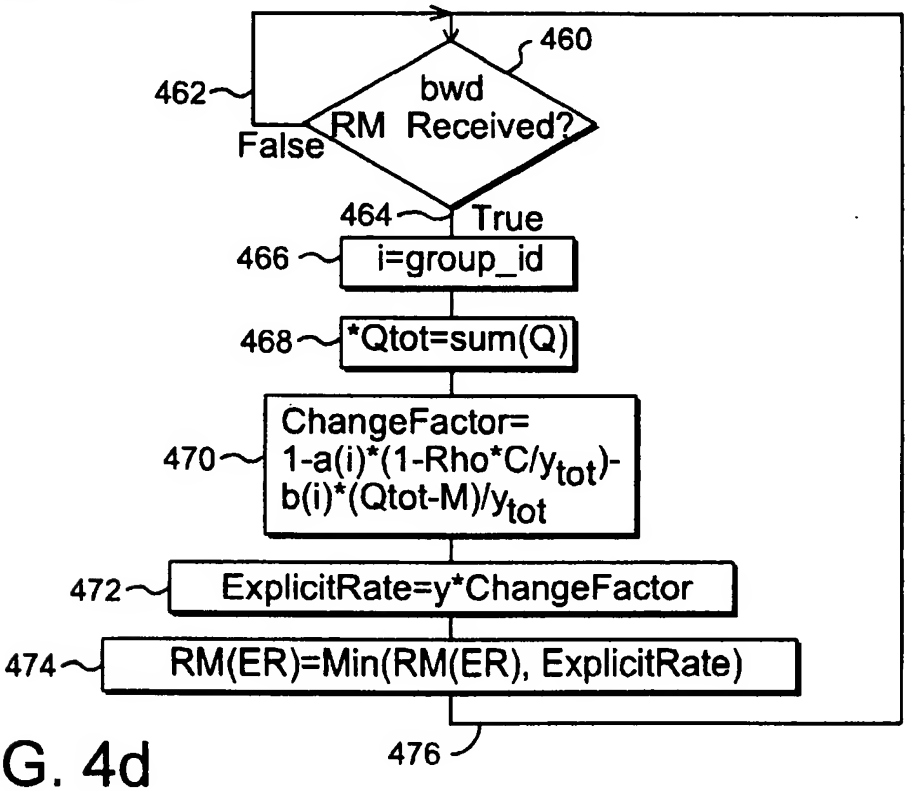

460
462
bwd
RM Received?
False
464
True
466
i=group_id
468
*Qtot=sum(Q)
470
ChangeFactor=
1-a(i)*(1-Rho*C/y_tot)-
b(i)*(Qtot-M)/y_tot
472
ExplicitRate=y*ChangeFactor
474
RM(ER)=Min(RM(ER), ExplicitRate)
476

FIG. 4d

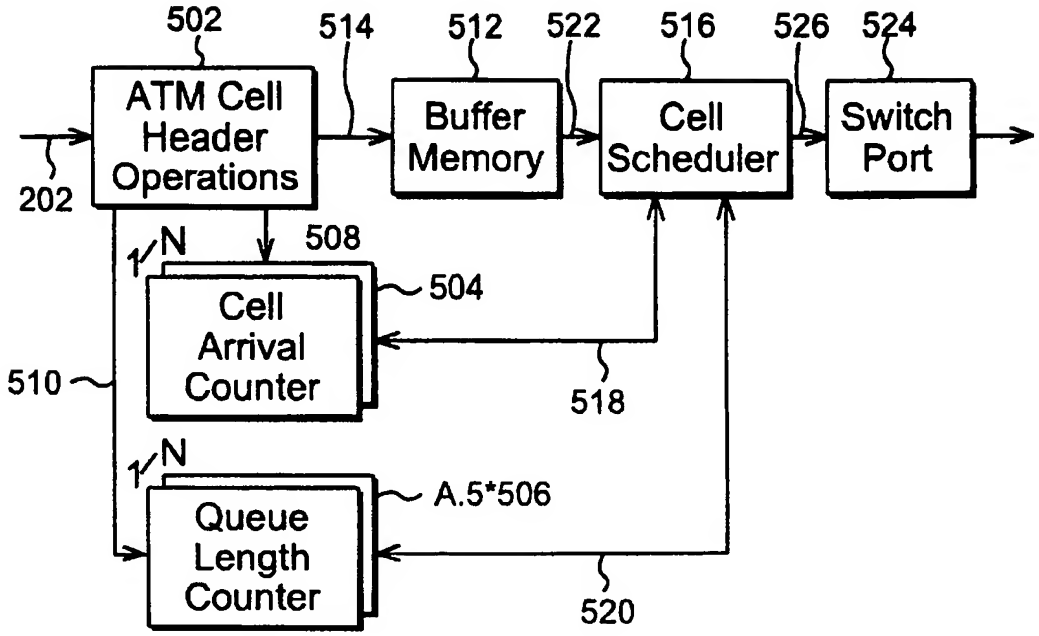

502
514
512
522
516
526
524
ATM Cell Header Operations
Buffer Memory
Cell Scheduler
Switch Port
202
1/N
508
504
Cell Arrival Counter
510
518
1/N
A.5*506
Queue Length Counter
520

FIG. 5

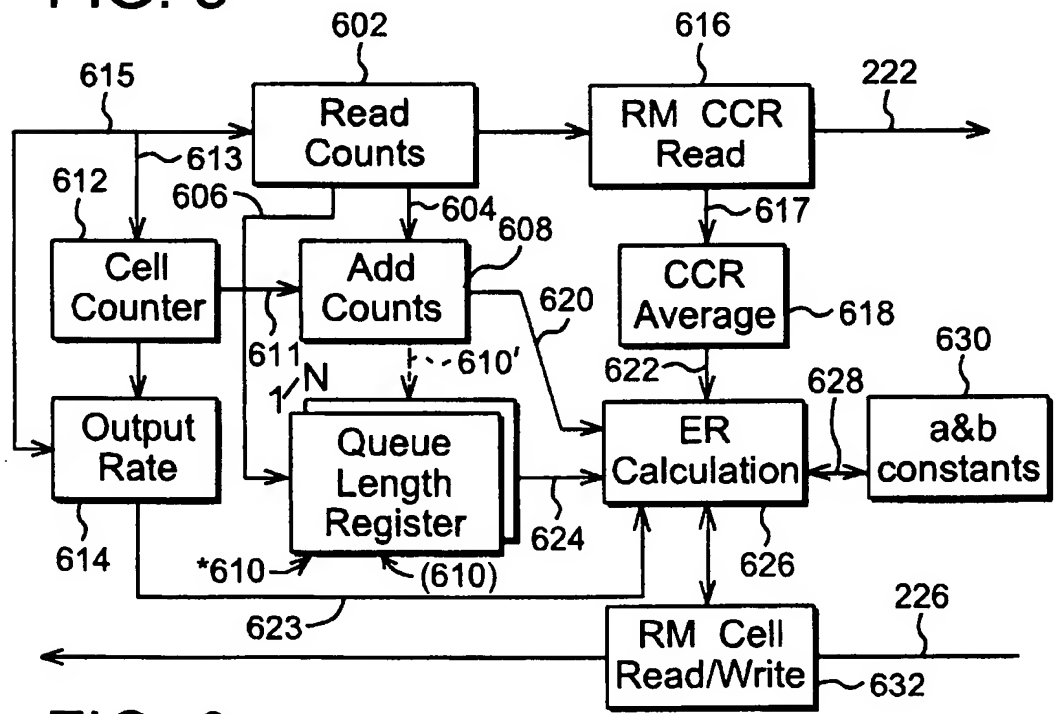

602
616
615
222
Read Counts
RM CCR Read
613
612
606
604
608
617
Cell Counter
Add Counts
CCR Average
618
620
611
610'
630
622
628
1/N
Output Rate
Queue Length Register
ER Calculation
a&b constants
614
624
626
*610
(610)
623
226
RM Cell Read/Write
632

FIG. 6

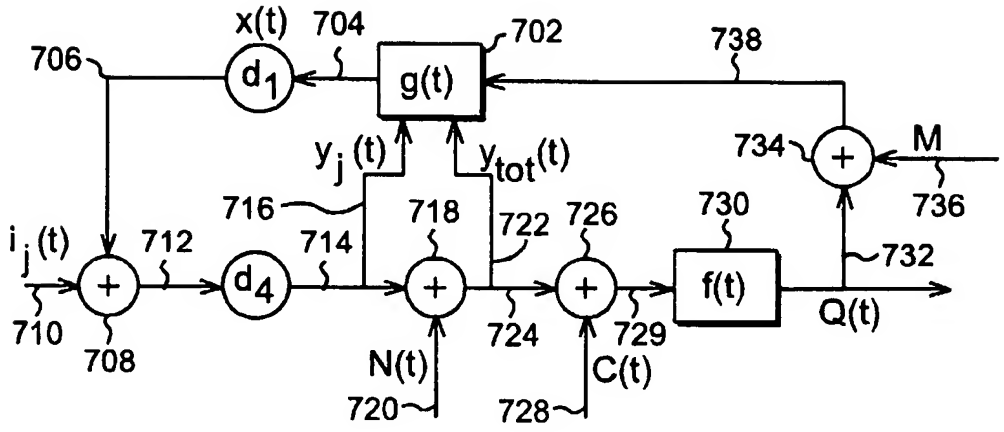

x(t)
704
702
738
706
d1
g(t)
yj (t)
ytot(t)
734
M
736
716
718
722
726
730
ij (t)
712
714
732
d4
f(t)
710
708
724
729
Q(t)
N(t)
C(t)
720
728

FIG. 7

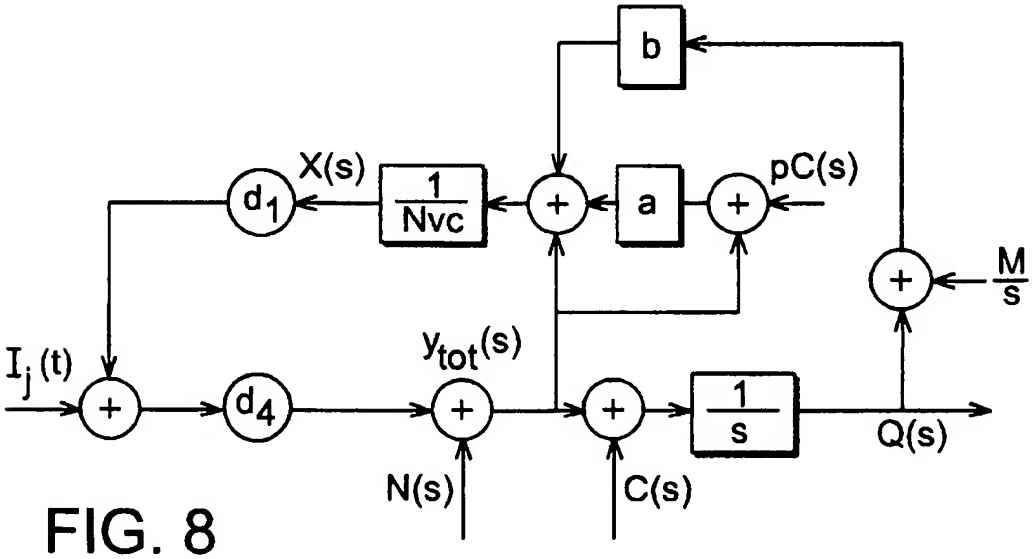

b
X(s)
pC(s)
d1
1/Nvc
a
M/s
Ij (t)
ytot(s)
d4
1/s
Q(s)
N(s)
C(s)

FIG. 8

FIG. 9

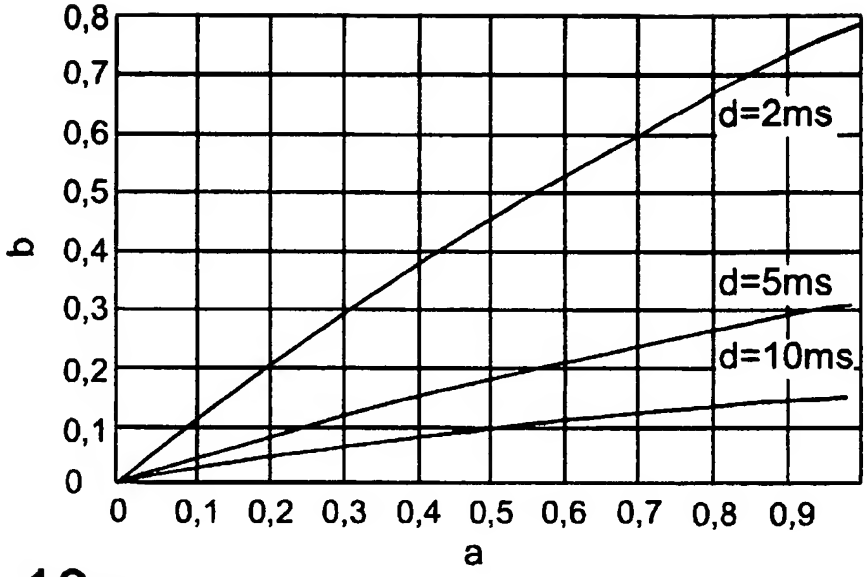


FIG. 10a

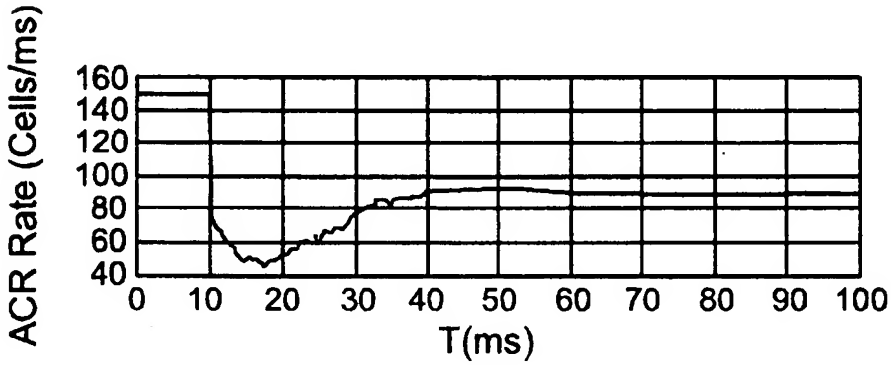


FIG. 10b

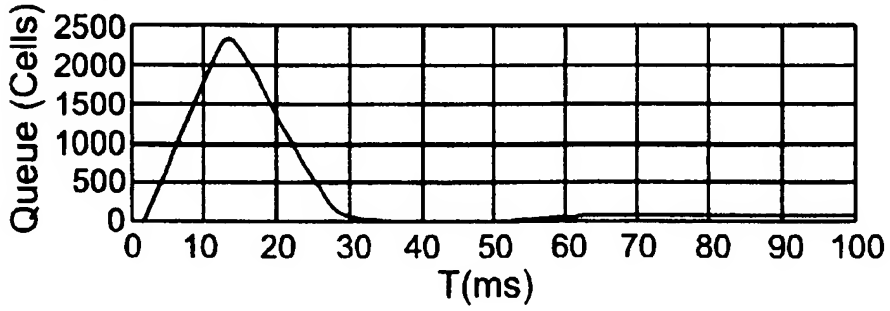


FIG. 11a

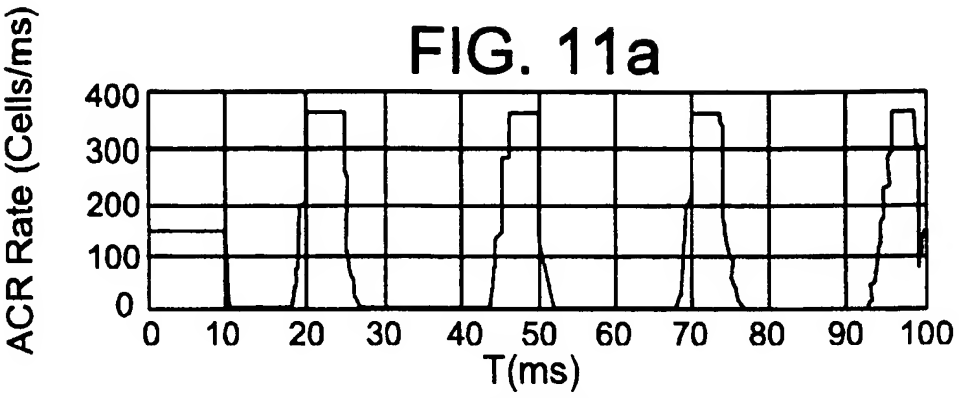


FIG. 11b

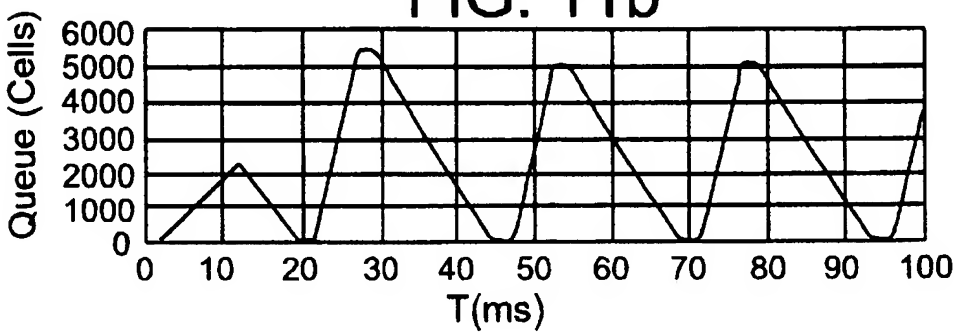


FIG. 12a

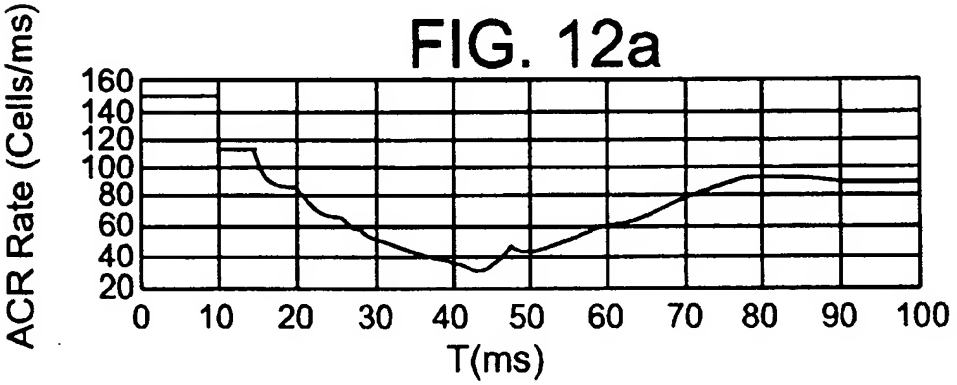
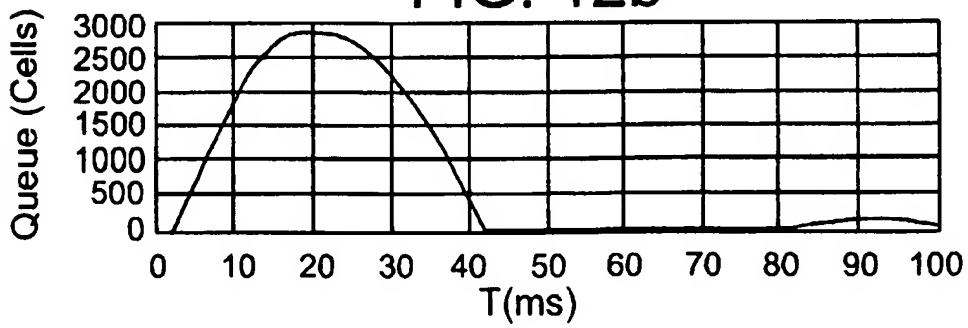


FIG. 12b



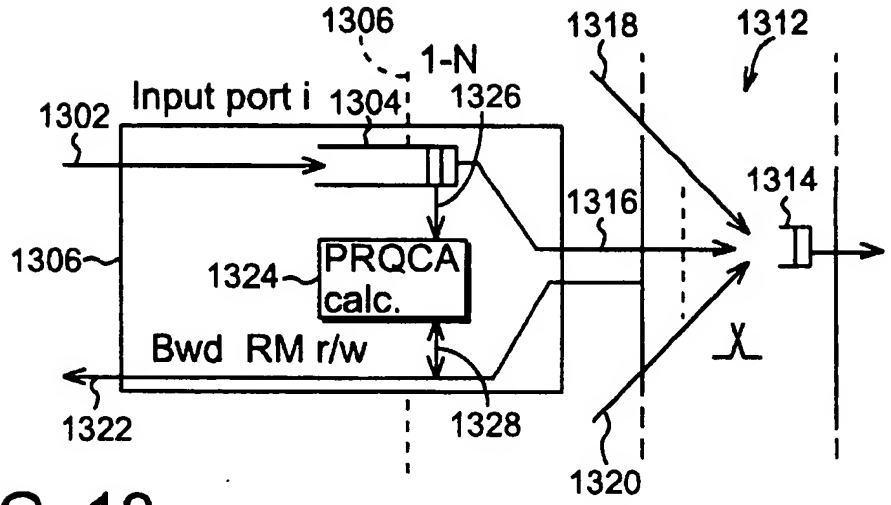

1306
1-N
1318
1312
Input port i
1304
1326
1302
1314
1316
1306
1324
PRQCA calc.
Bwd RM r/w
1322
1328
1320

FIG. 13

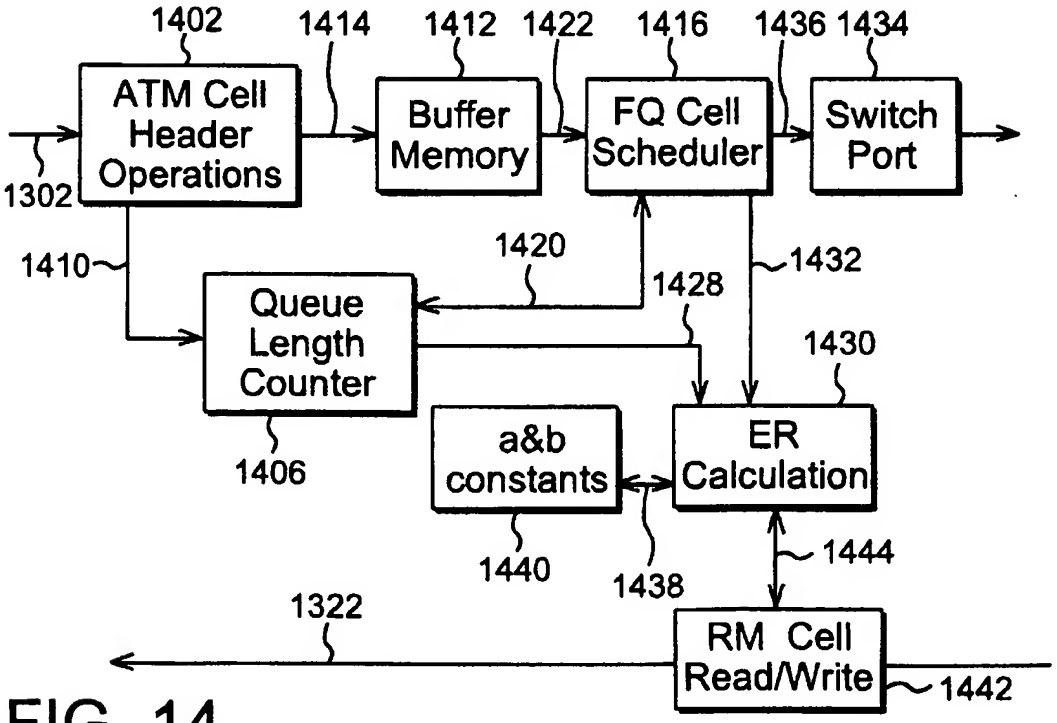

1402
1414
1412
1422
1416
1436
1434
ATM Cell Header Operations
Buffer Memory
FQ Cell Scheduler
Switch Port
1302
1410
1420
1428
1432
Queue Length Counter
1430
1406
a&b constants
ER Calculation
1440
1438
1444
1322
RM Cell Read/Write
1442

FIG. 14

FLOW AND CONGESTION CONTROL IN PACKET SWITCHED NETWORKS

This application is a continuation of International Application No. PCT/SE97/00439 filed on Mar. 14, 1997, which designates the United States.

This application is a continuation of International Application No. PCT/SE97/00439, which was filed on Mar. 14, 1997, which designated the United States, and which is expressly incorporated here by reference.

BACKGROUND

The present invention to a first aspect relates to a method and a system in an ATM system for controlling flows of data cells and flow control management cells from a number of sources to a destination over connections passing a network element, while returning the flow control management cells from the destination via the network element to the respective sources. The network element may be exposed to congestion due to contention between the connections, which necessitates queuing of the connections. The data cells include lower priority cells and higher priority cells. The flow control management cells have an explicit rate field for an explicit rate value used to limit a source maximum allowed cell rate to a specific value, and a current cell rate field for receiving said specific value.

According to a second aspect the invention relates to a method and a system in an ATM system for controlling flows of data cells and flow control management cells from a number of sources to a destination over connections passing a respective input buffer and a common output buffer of a fair queuing switch, while returning the flow control management cells from the destination via the switch to the respective sources. The switch is exposed to congestion due to contention between the connections, that necessitates queuing of the connections in the input buffers and output buffer. The flow control management cells have an explicit rate field for an explicit rate value used to limit a source maximum allowed cell rate to a specific value, and a current cell rate field for receiving said specific value.

The use of ATM as a new and overall solution for data communication, spanning from the local area to the wide area has proven to be burdened with a significant amount of problems. Most of these problems are more or less inherently connected to the properties of the data communication traffic as such, which to a great extent differ from characteristics well known from the telecommunications domain.

In general, applications using data communication services require large amount of bandwidth during rather short periods of time, causing offered traffic with bursty characteristics. Moreover, the information sent between typical computer applications must be error free, but may be affected by some transfer delay without deteriorated performance. Applications within the telecommunications domain could be said to have the opposite characteristics, i.e. the bandwidth is kept constant and they are not too sensitive to bit errors but sensitive to delays and variation of delays.

Consequently, data communication traffic must be managed differently from the telecommunication traffic within the ATM networks if the trade-off between network utilization and quality of service (QoS) is to be kept at an acceptable balance.

Recent efforts within standardization for a reflect the need for a specific ATM service to handle traffic with "pure" data communication properties. A service called the Available Bit Rate ABR service is specified in ATM Forum Traffic Management Specification 4.0, ATMF 95-0013R10, Feb. 1996. This service will be included in the ITU-T recommendations, as well, cf. e.g. ITU-T Recommendation I.371, "Congestion Management for the B-ISDN", 1995.

The ABR service utilizes a rate based congestion control concept. The network controls the rate at which the users mate transmit data by means of feedback information sent to the sources.

SUMMARY

A number of different explicit rate control algorithms have been proposed within the framework of the ABR service development. The complexity of these algorithms to a great extent depends on the choice of buffer scheduling principles in the switch. In order to limit the complexity, most of the algorithms so far have aimed at switches using FIFO scheduling. However, some of these algorithms use what could be called a "per VC accounting" to, for instance, keep track of the number of active connections in a switch and/or the number of stored cells per VC.

In most of the algorithms a buffer threshold is utilized in one way or the other to determine whether a switch is congested or not. The actions taken during congestion differ often quite drasticly from those taken during non congested conditions in order to alleviate the congestion condition. In some a bit more sophisticated algorithms the total input rate is compared to a desired rate reference, or workload, and a change in the input rate is ordered in proportion to the difference, cf. for example A. W. Barnhart, Hughes Network Systems, ATM Forum Contribution 95-0195, February 1995, "Example Switch Algorithm for Section 5.4 of TM Spec.". Simulations of the algorithm proposed in this document have indicated that when small values of a gain parameter is used, the algorithm has difficulties finding the accurate rate (fair rate) fast enough to efficiently use the unused bandwidth. When the gain parameter is given a higher value, the rate instead tends to oscillate and by that causes an unstable situation in the network. Another similar algorithm has been proposed by Raj Jain et. al., Ohio State University, ATM Forum Contribution 95-0195, October 1995, "ERICA+: Extensions to the ERICA Switch Algorithm".

It is an object of the present invention to provide an explicit rate mechanism that combines both information about the buffer occupancy and the input rate offered to a switch, and which can be used in a switch using an "ordinary" FIFO scheme as well as in a fair queuing scheduler.

This object has been attained by the method and the system according to the first and second aspects having attained the features appearing from claims **1–22** and **23–34** respectively.

In one important embodiment of the first aspect values of a set of parameters and variables are obtained, which include:

- y(t):a contention rate at the output buffer at time t,
- y_{tot}(t):a measured offered rate to the output buffer at time t,
- C(t):available rate at the buffer for lower priority cells at time t,
- Q(t):total queue length at the buffer at time t,
- p:fraction of an available rate at the buffer strived to,
- M:a buffer queue length reference,
- a_i and b_i: proportional constants for a connection i passing the output buffer.

Based upon these values the explicit rate value x_i(t) at time t for the connection i is calculated as

$$x_i t = y(t)\left[1 - a_i\left(1 - \frac{pC(t)}{y_{tot}(t)}\right) - b_i\left(\frac{O(t) - M}{y_{tot}(t)}\right)\right]$$

The explicit rate value $x_i t$ thus calculated is assigned to the explicit rate field of a backward flow control management, cell.

In one important embodiment of the second aspect values of a set of parameters and variables are obtained, which include:

$r_j(t)$:output rate at time (t) at the input buffer for connection j,

p(t):total load at time (t) on output from the input buffer calculated as the quotient number of APR cells received at the input buffer per time unit/number of occasions to send an ABR cell per time unit,

pref: desired load on output from the input buffer, $Q_j(t)$:queue length at time (t) at the input buffer for connection j,

M_j:queue length reference at the input buffer for connection j.

a and b:proportional constants for connection j,

Based upon these values the explicit rate value $x_j(t)$ at time t for the connection j is established as

$$x_j(t) = \max(r_j(t))\left[1 - a\left(1 - \frac{pref}{p(t)}\right) - b\left\{\frac{Q_j(t) - M_j}{\max(r_j(t))}\right\}\right]$$

wherein $\max(r_j(t))$ is an operation performed on the individual connection output rates rj(t) of the switch in order to find a common output rate for the connections passing the common output buffer and thus avoid a divergence in the individual connection rates. The explicit rate value $x_j t$ is assigned to the explicit rate field of a backward resource management cell passing the input buffer j.

By the invention the "lap time" on the control loop is shortened by catching information about the switch on the way back to the source. This state occurs after sending has first started with a delay caused by a full loop source-switch-destination.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will now be described more closely below with reference to the drawings on which

FIG. **1** is a schematic view of an ABR-connection extending in both directions between a source end system and a destination end system via a switch,

FIG. **2** is a view showing in more detail a part of the connection according to FIG. **1** extending between an input port and an output port of the switch, said input port and output port including an input device and an output device, respectively,

FIGS. **3***a–b* and **4***a–d* are flow diagrams illustrating operational steps performed in the input device and the output device illustrated in FIG. **2**,

FIGS. **5** and **6** are block diagrams illustrating in more detail the structure of the input device and output device of FIG. **2**, respectively,

FIGS. **7** and **8** show fluid flow approximations of an algorithm used for describing a system according to the invention as illustrated by means of FIGS. **1–6**,

FIG. **9** is a curve diagram illustrating the behaviour of two constants used in the algorithm according to the invention,

FIGS. **10***a,b* and **11***a,b* are curve diagrams illustrating the transient behaviour of the system according to the invention,

FIGS. **12***a,b* are curve diagrams illustrating the transient behaviour of a system operating according to a prior art algorithm.

FIG. **13** is a view similar to the one in FIG. **2** illustrating a switch in a further embodiment of the invention,

FIG. **14** in greater detail illustrates part of the switch of FIG. **13**.

DETAILED DESCRIPTION

FIG. **1** is a schematic view of an examplary connection in which the present invention may be used. More particularly, it is the question of an ABR end-to-end connection (ABR: Available Bit Rate) passing through a number of network elements, e.g. a number of switches, of which one is schematically indicated at **102**, in an ATM network. The switch **102** is the bottleneck along the connection and is assumed to be output buffered or logically having one common buffer per output port. According to the ATM Forum reference mentioned earlier, ABR is an ATM layer service category for which the limiting ATM layer transfer characteristics provided by a network may change subsequent to connection establishment. The service includes a flow control mechanism which supports feedback to control the cell transfer rate of the source rate in response to changing ATM transfer characteristics. Cell transfer rate in different locations along a connection such as the one forming an example here as usually expressed as cells/ms or Mbps and will be referred to below Just as rate. The feedback is conveyed to the source through specific control cells called Resource Management cells, or RM-cells. More particularly, in FIG. **1** ABR flow control occurs between a sending source end system **104**, below referred to as source, and a receiving destination end system **106**, below referred to as destination, representing a respective line termination and being interconnected via bi-directional connections. Per se, for a bi-directional ABR connection each connection termination point is both a source and a destination. However, for the sake of simplicity only the information flow from the source **104** to the destination **106** with its associated RM-cell flows will be considered here. Thus, by forward direction will here be meant the direction from the source **102** to the destination **104**, and the backward direction will be the direction from the destination **104** to the source **102**.

In FIG. **1** the source end system **104** is shown to be bi-directionally connected to a switch port, not shown, of the switch **102**. The bi-directional connection is represented by links **108** and **110** for the forward and backward direction, respectively. The links **108** and **110** are exposed to propagation delay represented by d_1 and d_4 within circles, respectively. The destination end system **106** is bi-directionally connected to the same switch port of the switch **102**. The bi-directional connection is represented by links **112** and **114** for the forward and backward direction, respectively. The links **112** and **114** are exposed to propagation delay represented by d_2 and d_3 within circles, respectively. At **116** and **118** output buffers ending in the links **112** and **110**, respectively, are indicated. Boxes denoted N represent other connections that use the same output buffer, i.e. the output buffers **116** and **118** are exposed to an aggregated cell flow from all active connections that can cause congestion at the buffers, this in turn causing a contention situation to occur between the connections in question. Transmission delay caused thereby from the buffers **116** and **118** is indicated by

symbols **120** and **122**. By congestion is here meant the same as defined in B-ISDN, viz. a state of network elements (e.g. switches, concentrators, cross-connects and transmission links) in which a network is not able to meet negotiated network performance objectives for already established connections and/or for new connection requests. In general, congestion can be caused by unpredictable statistical fluctuations of traffic flows and fault conditions within the network.

For the forward information flow from the source **104** to the destination **106** in FIG. **1**, there is a control loop consisting of two RM cell flows, one in the forward direction and one in the backward. The source **102** generates forward RM cells which are turned around by the destination **104** and sent back to the source as backward RM cells. These backward RM cells carry feedback information provided by the network elements and/or the destination back to the source. As stated in the ATM forum reference a network element may:

- Directly insert feedback control information into RM cells when they pass in the forward or backward direction.
- Indirectly inform the source about congestion by setting an explicit forward congestion indication bit in the data cell header of the cells of the forward information flow. In this case, the destination will update the backward RM cells based on this congestion information.
- Generate backward RM cells.

A box **124** in FIG. **1** represents various ABR related functions such as measurements at the buffers **116** and **118** of buffer queue length and/or offered rate, as well as RM cell read and write operations. The switch **102** may also insert backward RM cells to reduce feedback delays.

Feedback information from the switch **102** to the source **104** is conveyed in the backward RM cells at a rate proportional to the output rate. If an explicit rate scheme is used, the bottleneck rate for a connection is given by the minimum or explicit rates calculated in each switch or in the destination end system **106**. The explicit rate will here be referred to as ER.

The ER is given in an ER field of the RM cell and used to limit a source maximum allowed cell rate, denominated ACR, to a specific value. A field in the forward RM cell, denominated Current Cell Rate (CCR) is assigned the ACR at the source when sending away a forward RM cell. The ER is initially set by the source to a requested rate and may be subsequently reduced by any network element in the path to a value that the element can sustain.

Based on the description above with reference to FIG. **1**, the invention will now be elucidated by the use of an algorithm. The target system for the algorithm may be a network element, e.g. in the form of a physically or logically output buffered switch, such as the switch **102** in FIG. **1** with its output buffers **116** and **118**.

In order to shorten the feedback delay, the RM cells are assigned information in the reverse direction through the switch, which means that connections with the same delay in a loop including only source and switch, rather than in a loop including also the destination, will have the same control loop delay. The width of delay values within a group will be determined by the required tolerance of stability, on the one hand, and limitations due to implementation aspects, on the other hand. The algorithm uses two proportional constants, denoted a_i and b_i, respectively, for each group, correspondingly denoted G_i, of connections that have similar propagation delay $d=d_1+d_4$. The accuracy on how connections in practice are partitioned into groups is relaxed and delays within a group are assumed identical. The constants a_i and b_i are used as proportional constants for the measured error in rate and queue length, respectively. Rules to set these constants will be discussed further below. Optionally, all connections could be put in one and the same group of connections, which avoids a search for the appropriate pair of constants when a backward RM cell shall be given an ER value.

In the algorithm shown below, the explicit rate at time t for the group G_i is denoted $x_i(t)$ and calculated at the switch by the following formula:

$$x_i t = y(t)\left[1 - a_i\left\{1 - \frac{pC(t)}{y_{tot}(t)}\right\} - b_i\left\{\frac{Q(t) - M}{y_{tot}(t)}\right\}\right] \quad (1)$$

and assigned to the ER field of the RM cell, unless a lower ER value already was assigned. The algorithm uses the following variables:

- y(t):measured individual rate of contending connections at time t, also called contention rate henceforth,
- y_{tot}(t):measured offered rate to the buffer at time t,
- C(t):bandwidth capacity available at time t, also called available rate henceforth,
- Q(t):the buffer occupancy at time t, also called queue length henceforth,
- p:fraction of the available bandwidth that the algorithm tries to allocate,
- M:a queue length reference.

As discussed above, the CCR field in the forward RM cell is assigned the maximum allowed cell rate ACR at the source when a forward RM cell was sent away. The CCR is used to calculate the contention rate y(t) at the buffer and is referred to below as the fair share rate. The calculations are performed by means of an exponential averaging of the value in the CCR fields, i.e.

$$(2) y(t)=\alpha CCR_{con}+(1-\alpha)y(t),(2)$$

wherein

- α is an exponential averaging constant (in the interval [0,1]),

CCR_{con} denotes those CCRs encountered with a value equal or higher than $\beta y(t)$, where β is a fraction of the fair share rate that a CCR must exceed to be part of average.

This condition assures that only connections that actually have their bottleneck at the switch buffer in question, or at least have a rate close to the fair share rate, are taken into account and thereby avoids underflow of the buffer. The exponential averaging function (2) is commonly used in other FIFO based explicit rate ABR algorithms to derive a measure of the fair share rate and is then often denoted MACR calculation, cf. A. W. Barnhart, Hughes Network Systems, ATM Forum Contribution 95-0195, February 1995, "Example Switch Algorithm for Section 5.4 of TM Spec." In the algorithm proposed here the averaging is slightly modified to avoid starvation of updates if large immediate changes to the input rates occur. A counter counts the number of received underflowing CCR values and keeps the count. If the count exceeds a limit, the CCR value will be used anyway. Every reception of a CCR value from a contending connection will reset the counter.

The measurements of the offered rate y_{tot}(t) and the available rate C(t) are performed by taking ratios of cell counts and time intervals, defined by a minimum number of cells. C(t) is derived by taking the difference between the total link rate and bandwidth allocated by traffic with higher priority, i.e. variable bit rate VBR and continuous bit rate

CBR. In the VBR case this should be done within a relatively short time frame, while the CBR allocation only changes as connections are set up or released.

As stated above the algorithm is operating at the point in the switch where ABR connections are contending for the output link capacity, e.g. at an output buffer. However, for implementation reasons it may be difficult to locate sufficient buffering capacity at the output port. Instead, the buffers are located at the input side and in such a case the invention operates on a logical output buffer and use measures extracted from the distributed buffering structure.

Further below a case of location of the algorithm will be described. This location should be regarded as optional and alternative locations are fully possible with respect to the algorithm as such, but, however, limited for implementation reasons.

Even though the purpose of ABER is to utilize unused bandwidth, some bandwidth will be reserved static, e.g. for CBR traffic only, and must not be allocated by ABR. In a pseudo code to be described still further below, the static allocated bandwidth is always subtracted from the link rate.

A typical switch solution with distributed buffering has, as mentioned above, large input buffers at the input ports and relatively small buffers at the output ports. Cell losses in the small output buffer are avoided by means of an internal flow control mechanism, which does not form part of the invention. Furthermore, to avoid head of line blocking, the input buffers should be logically divided into separate buffers for each output port. The distribution of buffering means that the actual offered rate and queue will be spread between the input buffers, but the algorithm must operate with the total rate and queue length in order to cope with the fairness objectives. A way to attain this, schematically illustrated in FIG. **2**, is to let a switch internal cell format convey counts of arrived cells and measures or queue lengths from each logical input buffer to an output port, where the actual explicit rate calculation and backward RM cell assignment takes place.

FIG. **2** is a view showing in more detail a part of the connection according to FIG. **1** extending between an input port and an output port of the switch. In FIG. **2** an arrow **202** indicates cells arriving from the source, at a logical input buffer **204** of an input device $\mathbf{206}_i$. The input device $\mathbf{206}_i$ is a member of a number of input devices $\mathbf{206}_{1-N}$ belonging to the switch, here indicated at **212**. All of these input devices include logical input buffers, such as the buffer **204**. At each input device the total number of arriving cells and the number of queuing cells are counted to produce an arriving cell count Cy_i and a queuing cell count Q_i, respectively. Cells leaving the buffer **204** and entering an output buffer **214** in an output port, not shown, of the switch **212** are indicated by an arrow **216**. The output port including the buffer **214** is a member of N output ports belonging to the switch **212**. Arrows **218** and **220** indicate cell flows from the logical buffers of the other ones of the input devices $\mathbf{206}_{1-N}$ also entering the same switch output buffer **214**. Cells leaving the buffer **214** and the switch **212** are sent in the forward direction, indicated by an arrow **222**, via an output device **224** to a destination, not shown. The output device **224** is in common to the N output ports. Cells returned by the destination in the backward direction to the switch **212** via the output device **224** are indicated by an arrow **226**.

The input cell count and queue measurements, such as Q_i and Cy_i from the input device $\mathbf{206}_i$, are conveyed from each logical input buffer to the output device **224** in two fields of an internal cell format indicated at **228**. At the output device **224** the actual explicit rate calculation and backward RM cell assignment takes place in a function indicated by a block **230**, and a record of the total of arrived cell counts and queue length values for each of the input devices $\mathbf{206}_{1-N}$ must be maintained there. An arrow **232** pointing to the block **228** indicates the transfer of the Cy_i and Q_i values as well as read operation on forward RM cells. The delay caused by the output buffer **214** before the counts reach the output device **224** will be short compared to the overall propagation delay. The total offered rate will be calculated in continuous intervals and the instantaneous total queue length is simply the sum of the queue lengths conveyed from the input ports. When a backward RM cell is forwarded, the total queue value and the latest rate value is used to calculate the explicit rate value. A double arrow **234** between the arrow **226** and the block **228** indicates read and write operations on a backward RM cell.

A similar method to convey cell counts and queue length measurements as the one described above with reference to FIG. **2** may be used when large buffers are located at the output port. However no aggregation of the measurements from different input ports is necessary in this case. Instead the conveyed values may be used directly in the ER calculation. The operations performed in FIG. **2** will now be described below with reference to FIGS. **3–6**. FIGS. **3** and **4** contain flow diagrams illustrating operational steps performed in the in-put device **206** and output device **224**, respectively, and represented in pseudo code blocks following below. Arrivals of data or RM cells will be the events that clock the operations through different sequences. The description with reference to FIGS. **5** and **6**, showing parts of the structure of FIG. **2** in more detail, will further elucidate the performance of the operations.

The following variables and parameters will be used in the flow diagrams and the pseudo code:

NoRec :Total Count of received cells.

NoSentHP:Count of the number of higher priority cells sent with variable bit rate (VBR).

HPlimitstep:The maximum interval length for the output link rate calculation.

InputRateLimitstep:The maximum interval length for the input rate calculation.

StaticAlloc:Value of static allocated bandwidth for CBR traffic. It is provided by the signalling application or the management system for the network.

InputRateintLimit:Number of cells necessary for offered raze calculation.

OutputRateIntLimit:Number of cells necessary for output rate calculation.

Q:Vector of queue values for each logical input buffer.

Qtot:The total of the elements in Q.

Rho:Load Factor used for enabling control of the system towards a load between 0 and 1 in steady state.

β:Fraction of the fair share that a CCR must exceed to be part of average.

α:Exponential averaging constant (in the interval [0,1]).

a:Vector of rate proportional factors for each group of connections with similar propagation delay.

b:Vector of queue proportional factors for each group of connections with similar propagation delay.

The flow diagram in FIGS. **3***a* and **3***b* shows operational steps performed on ABR cells in the input device $\mathbf{206}_i$. FIG. **3***a* deals with ABR cell arrivals to the input device. Waiting for arrival of ABR cells is represented by block **302** and continue wait state arrow **303**. If an ASR cell is received, indicated by arrow **304**, cell count and queue length count

steps **306** and **308**, respectively, follow. The respective counts are indicated by Cy_i and Q_i in FIG. **3***a*. In the block indicating step **308** and on the corresponding line of the pseudo code (c1) below an asterisk indicates that this step is not used if an alternative queue length calculation is used, that will be described later. Step **308**, or step **306** in the alternative case, is followed by return to the wait state according to arrow **310**.

Code block (c1) below includes the state and operations shown in blocks **302**, **306** and **308**.

```
If Cell received
  (c1)Çyi:=Cyi+1
    Qi:=Qi+1
EndIf
```

Steps preparing for transmission of ABR cells to the output device **224** are shown in FIG. **3***b*. Waiting for transmission of ABR cells to the output device **224** is represented by block **312** and continue wait state arrow **313**. If an ABR cell shall be transmitted, indicated by arrow **314**, steps **316** and **318** follow. In these steps the values resulting from the operations of FIG. **3***a* are introduced into the respective two fields of the internal cell format indicated at **228** in FIG. **2**, followed by setting to 0 of Cy_i, step **316**, and reducing the Q_i count by 1, step **318**. In the block indicating step **318** and on the corresponding lines of the pseudo code (2) below an asterisk indicates that this stet is not used if the alternative queue length calculation is used. Step **318**, or step **316** in the alternative case, is followed by return to the wait state according to arrow **320**.

Code block (c2) below includes the state and operations shown in blocks **312**, **316** and **313**, the fields in question of the internal cell format being denominated "Cell_Cy_ field" and "Cell_Q_field":

```
If Cell due for transmission to output port
  Çell_Cy_field:=Cyi
  (c2) Cell_Q_field:=Qi
  Çyi:=0
  Qi:=Qi-1
EndIf
```

The flow diagram of FIG. **4***a* illustrates steps for calculating aggregated offered rate y_{tot} and queue length at the output device **224**. Waiting for transmission of ABR cells to output link is represented by a start step **402** and a return loop **403** containing an alternative start step **404**.

If an ABR cell shall be transmitted, steps **406** and **408** follow in which the contents Cy_i and Q_i of the arrival cell count and queue length fields **228** of the internal format are read and added to produce a total arrival rate count and a total queue length count Qtot, respectively. In the step **408** block and in code block (c3) below a queue length calculation expression following on an asterisk * indicates that this calculation is not used in case the alternative method to calculate the total queue length is used.

The counter values obtained via the Cy field in the internal cell format indicate the number of cells that have arrived to the switch input port containing the input buffer **204** (said input port being logically connected to the switch output port via the output device **224**) since the last cell departure. The counter value will be able to be zero if there are cells stored in the buffer **204** and no cells arrive to it. The same procedure is performed at all switch input ports so that the cells arriving to the input device **224** will bring with it the number of cells arriving between two arrivals to the respective logical input buffer. If these counter values are now summed the total number of cells arriving to a certain output port via a number of inputs will be obtained. If this value is divided with the time during which the counting is performed, there will be a total rate measure.

If, on the contrary, the number of arrivals is summed continuously and simultaneously this sum is decreased by 1 (Qtot) for each ABR cell that passes and delivers a Cy value, a net of the number of cells existing in the system for the time being will be received, i.e. the queue length. If the Cy values arriving are zero the sum will thus continuously decrease. As a matter of fact this is exactly the operation which would have been done for obtaining the queue length if there had been only one great output buffer, but in this case the counter values must be transferred to the output, which means a certain delay between the queue length calculation thus obtained and the contents of the input buffer queues at a certain arbitrary point in time.

The alternative way of counting the queue length is based upon the above considerations. If the alternative method is used, a queue counter at the input and registers used at the output for summing the individual queue lengths are not needed.

According to the alternative method the total queue length may be calculated at the output device purely based on the cell arrival count. More particularly, in the output device **224** a total queue length Qtot is calculated by summing the arrival cell counts Cy_i, counted in the input device **206** and conveyed to the output device **224** in the internal cell format **228** as described earlier, and subtracting therefrom the amount of 1 for each cell belonging to the controlled ABR cell flow, that is sent further on the output link to the destination in the step **408** block and in code block (c3) below this alternative calculation is expressed as [Qtot=max ((Qtot+Cell_Cy)-1,0)] where the decrementation -1 represents the cell arriving with the cell arrival count Cy, and 0 excludes negative Q values. The whole alternative expression is placed within brackets in order to indicate its character of alternative.

The alternative method implies that no separation between the values arriving from different input devices is necessary. Moreover, only one value has to be conveyed in the internal cell format. The alternative method could introduce some problems with synchronization between the actual number of stored cells and the measure Q in case of cells lost internally in the switch. However, this eventuality should be extremely exceptional if the switch is properly designed.

Step **410** determines when a predetermined number Input RateLimit of cells are received. If this is not the case, return to the start step **402** follows according to step **412**.

Step **404** in the return loop **403** establishes whether cells have not been received for a determined time period InputTtimelimit. This step is for obtaining a new value of y_{tot} in case cells should not arrive during a prolonged period of time. If any of the two conditions according to the respective steps **404** and **410** are fulfilled, step **414** follows, in which a time frame InputTdiff is calculated since last this step was operating. This time frame is used in step **436** following next, in which y_{tot} is calculated as the ratio between the total count of received cells and this time frame.

In the last step **418** the start time of a new time frame is set as the ending time of the preceding time frame, the count of received cells is set to zero, and a new starting point of time of the determined time period InputTimelimit is set. Return to the start step **402** then follows according to arrow **420**.

Code block (c3) below defines the operations of the flow diagram of FIG. **4***a*.

As stated earlier and follows from the above, $y_{tot}(t)$ is measured by taking ratio of cell count and time interval,

expressed below as "NoRec/InputTdiff" where the time interval "InputTdiff" is determined by "Now-InputTlast".

```
If ABRCell due for transmission on link
  NoRec:=NoRec+Cell_Cy_field
  Q(i):=Cell_Q_field
  [Qtot=max((Qtot+Cell_Cy)-1,0)]
  If NoRec>=InputRateIntLimit
    inputTdiff:=Now-InputTlast
    ytot:=NoRec/InputTdiff
    InputTlast:=Now
  (c3) InputTimeLimit:=Now+inputRateLimitstep
    NoRec:=0
EndIf
ElseIf (Now>InputTimeLimit)
  InputTdiff:=Now-InputTlast
  ytot:=NoRec/InputTdiff
  InputTlast:=Now
  InputTimeLimit:=Now+InputRateLimitstep
  NoRec:=0
Endif
```

The flow diagram of FIG. 4*b* illustrates operations in connection with calculation of bandwidth capacity available for ABR cells, i.e. available rate C(t), in the output device **224**.

Waiting for the appearance of high priority HP cells heading for our-put is represented by a starrt step **422** and a return loop **424** containing an alternative start step **426**. If an HP cell is received in step **422** an HP cell counting step **426** follows. The next step **430** establishes if more than a predetermined number OutputRateIntLimit of HP cells are received. If not, return to start step **422** follows according to arrow **432**.

Step **426** in the return loop **424** establishes whether HP cells have not been received for a determined time period HPtimelimit. This step is for obtaining a new value of C in case HP cells should not arrive during a long period of time. If any of the two conditions according to the respective steps **426** and **430** are fulfilled, step **434** follows, in which a time frame outputTdiff is calculated since last this step was operating. This time frame is used in step) **436** following next, in which the available rate (t) is calculated.

As stated earlier, the available rate C(t) is measured by taking the difference between the total link rate and bandwidth allocated by traffic with higher priority. This is expressed in the block **436** and in the following code block (c4) as "C:=LinkRate-StaticAlloc-NoSentHP/OutputTdiff". "LinkRate" is a configuration parameter that is provided by the management system for the network and states the type of rate provided by the layer below ATM. The meaning of "NoSentHP" and "StaticAlloc" has been explained earlier.

In the last step **438** the start time of a new time frame OutputTdiff is set as the ending time of the preceding time frame, the count of received cells in step **428** is set to zero, and a new starting point of time of the determined time period Hptimelimit is set. Return to the start step **422** then follows according to arrow **440**.

Code block (c4) below includes the states and operations shown in FIG. 4*b*.

```
If HPCell sent on output link
  NoSentHP:=NoSentHP+1
  If NoSentHP>=OutputRateIntLimit
  (c4) OutputTdiff:=Now-OutputTlast
    C:=LinkRate-StaticAlloc-NoSentHP/OutputTdiff
    OutputTlast:=Now
    HPtimelimit:=Now+HPlimitstep
    NoSent:=0
  EndIf
ElseIf (Now>HPtimelimit)
  OutputTdiff:=Now-OutputTlast
  C:=LinkRate-StaticAlloc-NoSentHP/OutputTdiff
  OutputTlast:=Now
  HPtimelimit:=Now+HPlimitstep
  NoSent:=0
EndIf
```

The flow diagram of FIG. 4*c* illustrates operations in connection with calculation of fair share rate y in the output device **224**.

Waiting for RM cell heading for output is represented by block **442** and continue wait state arrow **444**. If an RM cell is due for output, indicated by arrow **446**, step **448** establishes whether anyone of two conditions is fulfilled. The first one of these conditions is that CCRs have a value equal or higher than βy(t), expressed as "RM(CCR)>=(Beta*y) OR (UF_Count>UFlimit)", and the second condition is that a count, expressed as "UF_Count", of the number of received underflowing CCR values exceeds a limit expressed as "UFlimit". If any of the two conditions is fulfilled step **450** follows, otherwise step **452**.

In step **450** the operation expressed by formula (2) above is performed, this being expressed in pseudo code in FIG. 4*c* and in code block (c5) below as "y:=RM(CCR)*Alpha+y*(1-Alpha)". This is followed by step **454** in which UF_count is set to zero, and return to the wait state **442, 444** as indicated by arrow **456**.

In step **452** the operation expressed in pseudo code in code block (c5) below as "UF_count:=UF_Count+1" is performed, followed by return to the wait state **442, 444** according to arrow **458**.

Code block (c5) below includes the states and operations shown in FIG. 4*c*.

```
If Forward RM cell due for transmission on the output
  link
  If RM(CCR)>=(Beta*y) OR (UF_Count>UFlimit)
    y:=RM(CCR)*Alpha+y*(1-Alpha)
    (c5), UF_Count:=0
  Else
    UF_Count:=UF_Count+1
  EndIf
EndIf
```

FIG. 4*d* is a flow diagram illustrating the steps for explicit rate calculation. Waiting for arrival of RM cells is represented by a block **460** and continue wait state arrow **462**. If an RM cell is received, indicated by arrow **464**, determination of group G_i identity is made in step **466** for enabling choice of relevant constants a_i and b_i. Thereupon summation of all received queue values in step **468** follows if not the alternative queue length calculation is to be used, cf. the asterisk * in the step **468** block and in the pseudo code (c6) below.

In steps **470** and **472** operations for calculating explicit rate $x_i(t)$ according to formula (1) in the output device **24** follow. In step **470** a change factor for the contention rate y(t) is calculated, and in step **472** the explicit rate value is calculated by multiplying the change factor with the current contention rate y(t).

In step **474** the calculated explicit rate value is assigned to the ER field of a backward RM cell arriving to the switch, unless a lower ER value already was assigned to this ER field. This is expressed by the expression Min(RM(ER), ExplicitRate) included in the step **474** block. Steo **474** is followed by return to wait state **460, 462** according to arrow **476**.

The pseudo code (c6) below relates to the steps of FIG. 4*d*.

```
If Backward RM cell received
  i:=Group id for connection
  (c6) *Qtot:=Sum(Q)
  ChangeFactor:=1-a(i)*(1-Rho*C/ytot)-b(i)*Qtot-M/
    ytot
  ExplicitRate:=y*ChangeFactor
  RM (ER):=Min(RM(ER),ExplicitRate)
EndIf
```

FIG. **5** shows a more detailed block diagram over the input device **206**$_i$ in FIG. **2**. Cells arriving from the source, not shown, enter, according to the arrow **202** (FIG. **2**), a function **502** performing ATM cell header operations. The function **502** contains basic ATM functions such as VPI/VCI (Virtual Path Identifier/Virtual Channel Identifier) translation, PTI (Payload Type Indicator) operations, adding switch internal cell format. Blocks **504** and **506** represent a cell arrival counter and queue length counter, respectively, to be controlled by the ATM cell header operations function **502**, indicated by arrows **508** and **510**, respectively.

Referring to the flow diagram of FIG. **3***a*, the function **502** takes care of the wait state according to steps **302**, **303**, and the counters **504** and **506** perform the operations of steps **306** and **308**. As has been indicated earlier, the step **308** is not used if the alternative queue length calculation is used and therefore the counter **506** is superfluous in this case.

The cell arrival counter **504** consists of N separate counter registers, one for each of the N output ports of the switch **212**. For every arrived ABR cell a corresponding one of these counter registers is incremented according to step **306** and as expressed in line **2** of code block (c1) to produce the arriving cell count Cy$_i$. At transmission of the ABR cell the register is reset to zero as part of step **316** and as expressed by line **3** of code block (c2) above.

The queue length counter **506**, if used, comprises N separate counter registers, one for each of the N output ports of the switch **212**. For every arrived ABR cell a corresponding one of these counter registers is incremented according to step **308** and as expressed in line **3** of code block (c1) to produce the queuing cell count Q$_i$. At transmission of the ABR cell the register is decremented as part of step **318** and expressed in line **4** of code block (c2).

A buffer memory **512** receives, arrow **514**, the cells from the function **502** and stores these cells logically per output port to avoid head of line blocking. Higher priority will be given to cells from CBR and VBR connections, e.g. by implementing a head of line priority scheme.

A cell scheduler **516** is connected to the counters **504** and **506** for mutual control, as indicated by double arrows **518** and **520**, respectively. The cell scheduler **516** retrieves, indicated by arrow **552**, cells from the buffer memory **512** in such a way that there will always be a cell (CBR, VBR or ABR) in each cell time slot sent to the output port, originating from one of the logical input buffers. The only condition where this is not the case occurs if all output buffers are blocked simultaneously, i.e. the internal flow control stops the flow from all input devices. If a cell slot is allowed for transmission of an ABR cell to output port i, and in case of the alternative queue length calculation not being used, the current values Cy$_i$ and Q$_i$ of the cell arrival counter **504** register i and queue length counter **506** register i, respectively, are assigned to the respective fields of the internal cell format according to steps **316** and **318** in FIG. **3***b*. Upon each assignment the cell scheduler **516** resets the cell arrival counter **504** register i to zero and decrements the queue length counter **506** register i as part of these steps and described above. If the alternative queue length calculation is used, there is no queue length counter, and therefore the steps associated therewith are omitted.

A switch port **524** receives, arrow **526**, cells from the cell scheduler **516** and handles the transmission of every cell (CBR, VBR, ABR) into the switch core according to arrow **216** (FIG. **2**).

FIG. **6** is a block diagram illustrating the output device **224** and the function **230** of FIG. **2** in more detail. The cell flows **222** and **226**, cf. FIG. **2**, are included in FIG. **6**.

Reading of the cell count Cy$_i$ and queue length Q$_i$, the latter in case the alternative method of calculating queue length is not used, in the arriving internal format fields **228** (FIG. **2**) is indicated by a block **602** and arrows **604** and **606**, respectively. The read Cy$_i$ values are added according to step **406** in FIG. **4***a* and to line **2** in code block (c3) in an adding function **608** to form a total arrival rate count. The read values Q$_i$ according to line **3** in code block (c3) are stored in queue length registers **610**$_{1-N}$.

In the adding function **608** an aggregate count according to step **406** in FIG. **4***a* of all ABR cell arrivals headed for an output port is stored. When a predetermined number of cells are received, as determined by step **410** and calculated according to line **5** in code block (c3), a rate measure in the form of a ratio is calculated by the function **608** according to steps **414** and **416**, and lines **6** and **7** in code block (c3), i.e. y_{tot} is calculated. Alternatively, exceeding a maximum time value triggs the calculation as signalled according to arrow **611** from a cell counter **612**. The cell counter **612** counts, indicated by arrow **613**, the cells arriving synchronously in the flow **222** in the form of CBR, VBR, ABR or unassigned cells, and is used as a clock function to keep track of the maximum time for calculating each input rate y_{tot} by the function **608**.

The available output rate C for ABR cells is calculated by a function **614**. The function **614** counts, arrow **615**, the number of sent higher priority cells NoSentHP by identifying cells belonging to a VBR connection, and is connected to the cell counter **612** to keep track of the time period HPtimelimit. The function **614** also has access to the value of static allocated bandwidth StaticAlloc for CBR traffic.

The queue length values Q$_i$ are stored in one each of the registers **601**$_{1-N}$ and a total count is summed up from their contents, according to first line of the step **408** block and line **3** of code block (c3).

If the alternative method of calculating queue length is used, then the queue length registers **601**$_{1-N}$ and reading and summing queue length values Q$_i$ will be replaced by a single register **[610]** connected, according to dashed line arrow **610'**, to the adding function **608** for receiving the total count according to step **406** therefrom. The single register **[610]** will be updated according to the second line of the steo **408** block and line **4** of code block (c3).

The contents of the CCR field of every RM cell passing through the output device **224** in the flow **222** are read, indicated by block **616** and arrow **617**, by an averaging function **618**. The determination according to seep **448** in FIG. **4***c* is performed. Provided that any of the conditions in this step is fulfilled, an exponential averaging of the CCR values is performed in step **450** followed by step **454** in the function **618** in accordance with formula (2) and as expressed in lines **3** and **4** in code block (c5). This averaging is used as an approximation of the fair share rate of the switch **212** (FIG. **2**). If the conditions of step **448** are not fulfilled, step **452** replaces steps **450** and **454**.

When an ER value is to be calculated, the count and the offered rate value of the function **608**, the fair rate value of

the averaging function **618**, the available output rate C of the function **614**, and Qtot are available, indicated by arrows **620**, **622**, **623** and **624**, respectively, to an ER calculation function **626**. As follows from the above Qtot is calculated either as the summed up contents of all registers 610_{1-N} or, in case the alternative method is used, the updated contents of a single register [**610**].

The ER calculation is performed for each arrival of a backward RM cell as determined by step **460** in FIG. **4***d*. However, consecutive calculation could be based on the same value if RM cells arrive close in time. Depending on the group G_i of connections to which the RM cell belongs, as determined by step **466**, a pair of proportional constants a and 'b are retrieved, double arrow **628**, from a register **630**.

The available rate C(t) is calculated according to step **436** preceded by steps **422–434** and ended by step **438** in FIG. **4***b*, cf. also code block c(**4**). The calculation of the explicit rate is performed according to steps **470** and **472** preceded by step **468** in FIG. **4***d*.

A block **632** and a double arrow **634** to the ER calculation function **626** indicates the reading and writing of backward RM cells according to steps **460** and **474**, respectively. The block **632** may also include a function enabling insertion of new backward RM cells if required if the time interval between source originated RM cells becomes too long.

The transient behaviour of a system of the kind described above with reference to FIGS. **1–6** together with the formulas (1) and (2) will now be discussed. For this purpose fluid flow approximations of the system according to FIGS. **7** and **8** will be used. In FIG. **7** the system is shown in a block diagram form in which input and output signals are approximated by continuous functions. Some of the signals and functions have been explained in the previous subsections.

With respect to the input rate, or rate of queue length actually, the control algorithm according to formula (1) is a proportional and integrating (PI) algorithm where a is the constant for the proportional term and b the constant for the integrating term. Consequently, the queue length will be controlled by one term with the queue derivative corresponding to the proportional term, and one proportional term (PD) corresponding to the integrating term with respect to the rate. Thus, depending on what is studied the control is either working as a PI or PD algorithm.

In FIG. **7** a block **702** represents an explicit rate value calculating function producing as an output, flow arrow **704**, the explicit rate value x(t). The explicit rate value x(t) is sent back according to flow arrow **706** to the source end system. More particularly, the explicit rate value x(t) as affected by the propagation delay d_1 is superposed in an adding function **708** on a signal i_jt received at the adding function **708** according to flow arrow **710**. The result, according to flow arrow **712**, of the superposition is affected by the propagation delay d_4 to form the signal y_jt, flow arrow **714**.

The background of the signal i_jt is as follows. Each control loop will return the rate value that the sources shall use when sending in order to cope with congestion, i.e. x(t) of formula (1). It can, however, be suitable to introduce, in a model, deviations from the ideal input signal that the source may perhaps provide by e.g. sending at a slower rate or not at all. This deviation is modelled by means of the signal i(t) in order to introduce generally in the model. Furthermore, during the starting phase of connection there is no control loop established since no RM cells have been returned yet. In that case the sources send with the so called initial rate ICR.

The signal yj(t) represents the input rate from connection j and is approximated, in the system, by an exponential averaging according to formula (2). In the previous description of the algorithm no discretion on the input rates were made, thus yj(t) corresponds to y(t) in (1).

The signal y_jt forms an input to the function **702** according to arrow **716**, and to an adding function **718** to meet a signal N(t) forming another input to the adding function **718** according to arrow **720**. The signal N(t) models the traffic from other connections. If there are Nvc active connections then

$$N(t) = \sum_{j=1}^{Nvc-1} y_j(t) \tag{3}$$

The output flow, indicated by arrow **724**, from the adding function **718** is the signal y_{tot}(t) that forms another input according to arrow **722** to the function **702**, and an input according to an arrow **724** to a subtracting function **726**. The **30** bandwidth capacity or available rate C(t) is a further input according to arrow **728** to the subtracting function **726**.

In a fluid flow model, the buffer occupancy or queue length Q(t) can be expressed as the positive side of an integration of the total input rate y_{tot}(t) subtracted by the available rare C(t). The subtraction function **726** produces the difference y_{tot}(t)−C(t) that forms an input according to arrow **729** to an integrating function **730** that performs the integration of this difference, i.e.

$$Q(t)=\max\{(y_{tot}(t)-C(t)dt,0\}+Q(t_0) \tag{4}$$

The output Q(t) from the function **730** forms an input according to arrow **732** to a subtracting function **734** that as a further input receives the queue length reference M according to arrow **736** and produces the difference Q(t)−M. This difference forms according to arrow **738** a further input to the function **702**.

For positive queue values of the formula (4) the derivative of the queue length will be

$$\frac{d}{dt}Q(t) = y_{tot}(t) - C(t) \tag{5}$$

In the analysis of the system, the number of active and contending connections, Nvc, is assumed to be constant during the transients and all connections are assumed to have the same delays in the loop. Furthermore, the input rates per connections are also assumed to be equal, leading to the following approximation:

$$y_{tot}(t)=Nvc*y_j(t) \tag{6}$$

The explicit rate, x(t), cf. the formula (1), and the queue derivative can now be rewritten as

$$x(t) = y_j(t) - a\left\{y_j(t) - \frac{pC(t)}{Nvc}\right\} - b\left\{\frac{Q(t)-M}{Nvc}\right\} \tag{7}$$

and

$$\frac{1}{Nvc}\frac{d}{dt}Q(t) = y_j(t) - \frac{C(t)}{Nvc} \tag{8}$$

The control loop may now be closed according to FIG. **7**.

In order to give an explicit expression of the queue length the Laplace transforms of the continuous functions have

been utilized. The final expression for the queue length is given in equation

$$Q(s) = \frac{Nvc * I(s)e^{(d-d_1)s} + \{1 + a(p-1) - e^{ds}\}C(s) + b\frac{M}{S}}{se^{ds} - (1-a)s + b} \quad (9)$$

FIG. **8**, gives the control loop in a block diagram form using the Laplace transformed signals.

The zeros of the denominator of equation (9) will be crucial for the system behaviour and a stability analysis has also been performed to set guidelines for the setting of the parameters a and b. Given a certain value a (should be in the range 0,1) it can be shown that the following limit is valid for the b value to attain a stable system.

$$b < \frac{\sqrt{1-(1-a)^2}}{d} * \operatorname{atan}\frac{\sqrt{1-(1-a)^2}}{(1-a)} \quad (10)$$

Note however that this is valid for a system with equal delays only. A variety of delays will cause different behaviour and the setting of the a_i and b_i parameters in such cases should be done with necessary margins and be supported by simulations.

The expression above reveals that b must be scaled with the delay in the loop.

FIG. **9** is a diagram over the proportional constants a and b for different source-to-switch propagation delays d, the x-axis and y-axis showing a and b, respectively. An increase of the delay d leads to an inversely proportional descrease of **40** the constant b. Note that FIG. **9** shows the maximum values of b for each value of a that maintains an established control loop stable. However, when d is of the same order of magnitude as the time difference between two arrivals of RM backward cells, the accuracy of the linear model declines which results in a lower b value. Simulation shows that for a set to **0.8** values of b should be less than 1 to avoid oscillations.

FIGS. **10***a–b* and **11***a–b* show curves obtained as a result of simulations of a case, in a structure of the kind described above with reference to FIGS. **2–6**, where four sources are starting to transmit with an initial cell rate (ICR)=150 cells/ms (63.6 Mbps) each. FIGS. **10***a* and **11***a* on the y-axis show the allowed cell rate ACR in cells/ms. FIGS. **10***b* and **11***b* on the x-axis show time T in ms. in both simulations the source to switch propagation delay d was 5 ms and the rate proportional constant a was set to 0.8. In FIG. **10** the constant b is set to 0.1, which according to FIG. **9** is in a stable region for the control loop. As expected the transient does not exhibit any oscillations. The simulation curves in FIG. **11** have b=0.3 which gives an unstable control loop with undamped oscillations.

A comparative simulation of an algorithm disclosed in A. W. Barnhart, Hughes Network Systems, ATM Forum Contribution 95-0195, February 1995, "Example Switch Algorithm for Section 5.4 of TM Spec.", mentioned earlier, is presented in FIG. **12***a* and b which show curves in coordinate systems similar to those of FIG. **10***a*,**11***a* and **10***b*,**11***b*, respectively. The algorithm on which FIG. **12** is based gives a slower convergence towards the air share rate than the algorithm according to formula (1), but stays steady at the fair share when it is reached for the selected gain value (G=0.006). As indicated before, higher gain values cause undamped oscillation around the fair share rate.

The description thus far above presupposes FIFO scheduling only, but the algorithm may be modified to work in a fair queuing context as well.

In essence fair queuing, denominated FQ below, aims to provide equal sharing of the output resources of a switching device e.g. the output link bandwidth. Opposed to a FIFO scheduling scheme, traffic arriving to a FQ switch may be served earlier than traffic that arrived at an earlier point in time. A straight forward way to implement a FQ scheme is to have individual queues for each connection in the switch and serve these cyclically, giving each connection (session) a time slice of service each. In particular, for an ATM network such a time slice would be one cell i.e. one cell from each active ATM connection is sent in a cyclic manner, also referred to as "round robin scheduling". This type of mechanism avoids conditions where a bursty source allocates the entire output capacity for long time periods, which may happen in a FIFO case. For more details regarding FQ, reference is made to the reports stated below in which different alternative methods to accomplish fair queuing are described and analysed.

S. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications", Globecom 1994.

K. Parekh, R. Gallager, "A generalized Processor Sharing Approach to Flow Control in integrated Services Network: The Single Node Case", IEE/ACM Transactions on Networking, Vol. 1, No 3, June 1993.

British patent application . . . , corresponding to EUA's document X95 5181, "Distributed Weighted Fair Queuing".

The main difference, as compared with a FIFO scheduling scheme, when using the invention in a fair queuing (FQ) switch is that it is possible to control the occupancy in the individual VC queues. Moreover, it is possible to derive the actual fair share rate of the switch by measuring the output rate of a connection. This is, however, only valid if there are cells in the queues, i.e. the load of the switch is 1. It is therefore important that the explicit rate value sent back to the source end system is related to the current fair share of the switch. Otherwise some connections will get larger portions of bandwidth than others, since round robin scheduling will not result in fair allocation of link capacity if there are not cells in all queues all the time.

By load is here and conventionally meant actual output rate from a system in relation to the maximal output rate of the system, i.e. If cells are sent in to the system with a rate equal to or higher than that of the output link, the load will be 1 (or 100%). In the fair queuing case, to be described more closely below, it is not the question of considering the total input rate (ytot) to the system and relating this to the link rate in, "change factor" as in the FIFC case. Using traffic theoretical terms, carried traffic is considered in the fair queuing case, and offered traffic is considered in the FIFO case (the concept traffic is measured in Erlang and is essentially always used in switch contexts as a measure of capacity).

With some minor modifications, the same model of the control loop as for the FIFO case may be used in the FQ case also. The signal N(t) is set to zero, because no other connections are using the queue. The effect of the other active connections are instead observed in the available rate which will vary linearly with the number of active connections. The resulting output capacity for each connection will be affected by the number of other active ABR connections (FQ operates on each connection) plus of course VBR and CBR influence.

As just mentioned, offered traffic is no longer used as part of the change factor, but now carried traffic or, in other words, the load on the system is used.

In the version of the invention relating to the FQ switch the explicit rate is calculated according to

$$x(t) = \max(r_j(t))\left[1 - a\left(1 - \frac{pref}{p(t)}\right) - b\left\{\frac{Q_j(t) - M_j}{\max(r_j(t))}\right\}\right] \quad (11)$$

where $r_j(t)$ is the fair share or output rate for connection j, and p(t) is the total load on the output link calculated as the quotient: number of sent AER cells per time unit/the number of occasions to send an ABR cell per time unit. The load will thus be 1 if all occasions of allowance of sending a cell are used. Instead of measuring the input rate ytot as in the FQ case, the load on the buffer, p(t), is measured, which corresponds to the carried traffic. In the formula above this measure is related to a reference load pref, being the desired load on the system. When the load t(t) is lower than the reference load the term will contribute with a positive value to increase the load. The queue length will be measured on a per connection basis which is possible in a FQ environment. The max operation performed on the individual connection output rates rj(t) is done in order to find the common fair share rate of the switch and thus avoid a divergence in the individual connection rates. The reason for relating change factor to a value for all connections is that there is a risk that the connections strive towards different rates, i.e that the change factor value moves towards 1 but the associated connections have landed on different rates. In other words, although the equation systems have several different solutions for steady state, the one in which all rates are equal is desirable here.

An embodiment of ER calculation for a FQ scheme where all operations take place at the input port will now be described with reference to FIGS. **13** and **14**.

FIG. **13** is a schematic view, similar to FIG. **2**, of the input port showing part of a two-way connection extending through a switch between a source and destination. In FIG. **13** an arrow **1302** indicates cells arriving from the source, not shown, at a logical input buffer **1304** of the input port, designated 1306_i. The input port 1306_i is a member of a number of input ports 1306_{1-N} belonging to the switch, indicated at **1312**. All of these input ports include logical input buffers, such as the buffer **1304**. At each input port the number of queuing cells are counted to produce a queuing cell count Q_i. Cells leaving the buffer **1304** and entering an output buffer **1314** located in an output port, not shown, of the switch **1312** are indicated by an arrow **1316**. The output port including the buffer **1314** is a member of N output ports belonging to the switch **1312**. Arrows **1318** and **1320** indicate cell flows from the logical buffers of the other ones of the input ports 1306_{1-N} also entering the same switch output buffer **1314**.

Cells returned by the destination in the backward direction through the switch **1312** and the input port **1306** are indicated by an arrow **1322**.

In the input port **1306** the actual explicit rate calculation and backward RM cell assignment takes place in a function indicated by a block **1324**. An arrow **1326** pointing to the block **1324** indicates transfer of the Q_i count. A double arrow **1328** between the arrow **1322** and the block **1324** indicates read and write operations on a backward RM cell.

FIG. **14** shows a more detailed block diagram over the input port 1306_i in FIG. **13**. Cells arriving from the source, not shown, enter, according to the arrow **1302** (FIG. **13**), a function **1402** performing ATM cell header operations. The function **1402** contains basic ATM functions such as VPI/VCI (Virtual Path Identifier/Virtual Channel Identifier) translation, PTI (Payload Type Indicator) operations, adding switch internal cell format. A block **1406** represents a queue length counter to be controlled by the ATM cell header operations function **1402**, indicated by arrow **1410**.

The queue length counter **1406** is incremented for every arrived ABR cell to produce the queuing cell count Q_i. At transmission of the ABR cell the register is decremented.

A buffer memory **1412** receives, arrow **1414**, the cells from the function **1402** and stores these cells logically per output port to avoid head of line blocking. Higher priority will be given to cells from CBR and VBR connections, e.g. by implementing a head of line priority scheme.

A cell scheduler **1416** is connected to the counter **1406** for mutual control, as indicated by double arrow **1420**. The cell scheduler **1416** retrieves, indicated by arrow **1452**, cells from the buffer memory **1412** in such a way that there will always be a cell (CBR, VBR or ABR) in each cell time slot sent to the switch output port, originating from the logical input buffer **1304**. The cell scheduler **1416** is coordinated with the corresponding cell schedulers of all inputs, implying that it can count the time T for each new "sweep" over the connections that the FQ algorithm does (one cell from each connection is sent out). 1/T then provides the fair share rate $r_j(t)$ that the system has at the moment towards a certain output. Furthermore the load p(t) on the output link **1316** is calculated as the quotient: number of sent ABR cells per time unit/the number of occasions to send an ABR cell per time unit, as has been mentioned earlier. This is done separately for each input buffer **1304**, but if the FQ algorithm works as it should, this provides a measure of the total load on the output from the output buffer **1314**.

When an ER value is to be calculated, the count on the queue length counter is available, indicated by arrow **1428**, to an ER calculation function **1430**. The values of the load p(t) and the fair share rate $r_j(t)$ are available, arrow **1432**, on the FQ cell scheduler **1416**.

The ER calculation is performed for each arrival of a backward RM cell **1322**. A switch port **1434** receives, arrow **1436**, cells from the cell scheduler **1416** and handles the transmission of every cell (CBR, VBR, ABR) into the switch core according to arrow **1316** (FIG. **13**).

A pair of proportional constants a and b are retrieved, double arrow **1438**, from a register **1440**. A block **1442** and a double arrow **1444** to the ER calculation function **1430** indicates the reading and writing of backward RM cells. The block **1442** may also include a function enabling insertion of new backward RM cells if required, if the time interval between source originated RM cells becomes too long.

Also in the FQ case the algorithm can be applied to a system allowing the use of great output buffers, meaning that the coordination of cell schedulers can be dispensed with. In this case the counter values must be transferred to the output as for the FIFO case, which means a certain delay between the queue length calculation at the output and the contents of the input buffer queues at a certain arbitrary point of time.

What is claimed is:

1. A method in an ATM system for controlling flows of data cells and flow control management cells from a number of sources to a destination over connections passing a network element, while returning the flow control management cells from the destination via the network element to their respective sources,

said network element being exposed to congestion due to contention between the connections, said contention necessitating queuing of the connections,

said data cells including lower priority cells and higher priority cells, and said flow control management cells having an explicit rate field for an explicit rate value used to limit a source maximum allowed cell rate to a specific value, and a current cell rate field for receiving said specific value,

comprising performing in the network element the steps of

counting a predetermined number of higher priority cells due to be sent on an output link from the network element to the destination while keeping track of the time interval taken for performing the

counting to produce a higher priority cell rate in the form of a ratio between the counted higher priority cells and the time interval,

calculating an available rate value for lower priority cells as a difference between a total available link rate value and the higher priority cell rate,

establishing a queue length reference forming a desirable queue length,

calculating deviations from the available rate value and the queue length reference due to receiving varying amounts of cells on contending connections,

calculating a modified explicit rate value as a function of these deviations, and

introducing the modified explicit rate value into the explicit rate field of the backward flow control management cells.

2. The method according to claim **1**, comprising performing in the network element further steps of

counting cells arriving from the respective sources to produce source specific cell counts,

counting the source specific cell counts to produce a determined total count of received cells while keeping track of the time interval taken for performing the counting to produce an offered rate value as a ratio between the determined total count value and a time interval,

calculating a total queue length for all arriving cells,

calculating the deviation from the available rate value as a difference between the available rate value and the offered rate value,

calculating the deviation from the queue length reference as a difference between the total queue length and the queue length reference.

3. The method according to claim **2**, comprising performing in the network element the further steps of

keeping track of a first condition implying that received current cell rate field values are equal to or higher than a threshold value and a second condition implying that a limit number of received current cell rate field values lower than the threshold value has been exceeded,

calculating, provided that any of these conditions is fulfilled, a contention rate as an exponential averaging of the current cell rate field values,

performing the calculation of the explicit rate value while using the contention rate as a multiplication factor.

4. The method according to claim **2** or **3**, comprising performing in the network element a further step of multiplying the deviations from the available rate value and the queue length value each by a proportional constant.

5. The method according to claim **4**, comprising performing in the network element the further step of determining the values of respective constants of each pair of proportional constants a and b for each connection j as depending upon the propagation delay in a loop extending over the source and the network element.

6. The method according to claim **2**, comprising performing in the network element the further step of decreasing the available rate value by means of a multiplication factor.

7. The method according to claim **2**, comprising performing in the network element further steps of counting queuing cells arriving from the respective sources to produce source specific queuing cell counts, and calculating the total queue length by summing all source specific queuing cell counts.

8. The method according to claim **2**, comprising performing in the network element further steps of counting cells arriving from the respective sources to produce source specific cell counts, calculating the total queue length by summing these counts and subtracting therefrom the amount of 1 for each cell that is sent on the output link.

9. The control system in an ATM system for controlling flows of data cells and flow control management cells from a number of sources to a destination over connections passing a network element, while returning the flow control management cells from the destination via the network element to their respective sources,

said network element being exposed to congestion due to contention between the connections, said contention necessitating queuing of the connections,

said data cells including lower priority cells and higher priority cells, and said flow control management cells having an explicit rate field for an explicit rate value used to limit a source maximum allowed cell rate to a specific value, and a current cell rate field for receiving said specific value,

comprising

means for counting a predetermined number of higher priority cells due to be sent on an output link from the network element to the destination while keeping track of the time interval taken for performing the counting to produce a higher priority cell rate in the form of a ratio between the counted higher priority cells and the time interval, and

means for calculating an available rate value for lower priority cells as a difference between a total available link rate value and the higher priority cell rate,

establishing a queue length reference forming a desirable queue length,

calculating deviations from the available rate value and the queue length reference due to receiving varying amounts of cells on contending connections,

calculating a modified explicit rate value as a function of these deviations, and

introducing the modified explicit rate value into the explicit rate field of the backward flow control management cells.

10. The system according to claim **9**, comprising

means for counting cells arriving from the respective sources to produce source specific cell counts,

means for counting the source specific cell counts to produce a determined total count of received cells while keeping track of the time interval taken for performing the counting to produce an offered rate value as a ratio between the determined total count value and a time interval, and

means for

calculating a total queue length for all arriving cells,

calculating the deviation from the available rate value as a difference between the available rate value and the offered rate value,

calculating the deviation from the queue length reference as a difference between the total queue length and the queue length reference.

11. The system according to claim **10**, comprising means for

keeping track of a first condition implying that received current cell rate field values are equal to or higher than a threshold value and a second condition implying that a limit number of received current cell rate field values lower than the threshold value has been exceeded,

calculating, provided that any of these conditions is fulfilled, a contention rate as an exponential averaging of the current cell rate field values,

performing the calculation of the explicit rate value while using the contention rate as a multiplication factor.

12. The system according to claim **9** or **10**, comprising means for multiplying the deviations from the available rate value and the queue length value each by a proportional constant.

13. The system according to claim 12, comprising means for determining the values of respective constants of each pair of proportional constants a and b for each connection as depending upon the propagation delay in a loop extending over the source and the network element.

14. The system according to claim 10, comprising means for performing in the network element the further step of decreasing the available rate value by means of a multiplication factor.

15. The system according to claim 10, comprising means for counting queuing cells arriving from the respective sources to produce source specific queuing cell counts, and calculating the total queue length by summing all source specific queuing cell counts.

16. The system according to claim 10, comprising means for counting cells arriving from the respective sources to produce source specific cell counts, calculating the total queue length by summing these counts and subtracting therefrom the amount of 1 for each cell that is sent on the output link.

17. A method in an ATM system for controlling slows of data cells and flow control management cells from a number of sources to a destination over connections passing an output buffer of a switch, while returning the flow control management cells from the destination via the switch to their respective sources,

said output buffer being exposed to congestion due to contention between the connections, said contention necessitating queuing of the connections,

said data cells including lower priority cells and higher priority cells, and said flow control management cells having an explicit rate field for an explicit rate value used to limit a source maximum allowed cell rate to a specific value, and a current cell rate field for receiving said specific value,

comprising performing the steps of

obtaining values of a set of parameters and variables including

y(t): a contention rate at the output buffer at time t,

$y_{tot}(t)$: a measured offered rate to the output buffer at time t,

C(t): available rate at the buffer for lower priority cells at time t,

Q(t): total queue length at the buffer at time t,

p: fraction of an available rate at the buffer strived for,

M: a buffer queue length reference,

a_i and b_i: proportional constants for a connection i passing the output buffer,

determining the explicit rate value $x_i(t)$ at time t for the connection i as $x_it=y(t)[1-a_i\{1-pC(t)/y_{tot}(t)\}-b_i\{Q(t)-M/y_{tot}(t)\}]$, where i is an integer, and

assigning the explicit rate value x_it to the explicit rate field of a backward flow control management cell.

18. The method according to claim 17, comprising

counting cells arriving from the respective sources to produce source specific cell counts,

counting the source specific cell counts to produce a determined total count value of received cells while keeping track of the time interval taken for performing the counting to produce the offered rate value $y_{tot}(t)$ as a ratio between the determined total count value and the time interval.

19. The method according to claim 18, comprising

keeping track of a first condition implying that received current cell rate field values are equal to or higher than a threshold value and a second condition implying that a limit number of received current cell rate field values lower than the threshold value has been exceeded,

calculating, provided that any of these conditions is fulfilled, the contention rate (y(t)) as an exponential averaging of the current cell rate field values.

20. The method according to claim 19, comprising determining the values of the respective constants a_i and b_i as depending upon the propagation delay in a loop extending over the switch and the source from which the connection i starts.

21. The method according to claim 18, comprising counting queuing cells arriving from the respective sources to produce source specific queuing cell counts, and calculating the total queue length by summing all source specific queuing cell counts.

22. The method according to claim 18, comprising counting cells arriving from the respective sources to produce source specific cell counts, and calculating the total queue length by summing these counts and subtracting therefrom the amount of 1 for each cell that is sent on the output link.

23. A method in an ATM system for controlling flows of data cells and resource management cells from a number of sources to a destination over connections passing a respective input buffer and a common output buffer of a fair queuing switch, while returning the resource management cells as backward resource management cells from the destination via the switch to the respective sources,

said switch being exposed to congestion due to contention between the connections, said contention necessitating queuing of the connections in the input buffers and the output buffer,

said resource management cells having an explicit rate field for an explicit rate value used to limit a source maximum allowed cell rate to a specific value, comprising

obtaining values of a set of parameters and variables including

$r_j(t)$: output rate at time (t) at the input buffer for connection j,

p(t): total load at time (t) on output from the input buffer calculated as the quotient number of ABR cells received at the input buffer per time unit/number of occasions to send an ABR cell per time unit,

pref: desired load on output from the input buffer,

$Q_j(t)$: queue length at time (t) at the input buffer for connection j,

M_j: queue length reference at the input buffer for connection j,

a and b: proportional constants for connection j,

determining the explicit rate value $x_j(t)$ at time t for the connection j as

$$x_j(t)-\max(r_j(t))[1-a\{1-\mathrm{pref}/p(t)\}-b\{Q_j(t)-M/\max(r_j(t))\}]$$

wherein $\max(r_j(t))$ is an operation performed on the individual connection output rates $r_j(t)$ of the switch in order to find a common output rate for the connections passing the common output buffer and thus avoid a divergence in the individual connection rates, and j is an integer, and

assigning the explicit rate value x_{jt} to the explicit rate field of a backward resource management cell passing the input buffer j.

24. The method according to claim 23, comprising determining the values of the constants a and b as depending upon the propagation delay in a loop extending over the source and the switch.

* * * * *